aws

Guía del usuario

Amazon ECR

Versión de API 2015-09-21

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon ECR: Guía del usuario

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon ECR? 1
Conceptos y componentes 1
Casos de uso comunes 3
Características de Amazon ECR 5
Primeros pasos con Amazon ECR 6
Precios de Amazon ECR 6
Desplazamiento de una imagen a lo largo de su ciclo de vida 7
Requisitos previos 7
Instale el AWS CLI 7
Instalar Docker 7
Paso 1: creación de una imagen de Docker 9
Paso 2: crear un repositorio 11
Paso 3: autenticar en su registro predeterminado 12
Paso 4: Insertar una imagen en Amazon ECR 12
Paso 5: Extraer una imagen de Amazon ECR 14
Paso 6: eliminar una imagen 14
Paso 7: eliminar un repositorio 15
Optimización del rendimiento 16
Realizar solicitudes 18
Empezando con IPv6 19
Probar la compatibilidad de dirección IP 19
Realizar solicitudes con puntos de enlace de doble pila 20
Uso de puntos de enlace de Amazon ECR desde la CLI de docker 21
Uso de IPv6 direcciones en las políticas de IAM 21
Registro privado 23
Conceptos sobre el registro 23
Autenticación del registro 23
Uso del ayudante de credenciales de Amazon ECR 24
Uso de un token de autorización 24
Uso de la autenticación de la API HTTP 25
Opciones del registro 26
Permisos de registro 27
Ejemplos de políticas de registro 28
Cambiar al ámbito ampliado de la política de registro 31

Concesión de permisos para la replicación entre cuentas ... 33
Concesión de permisos para caché de extracción ... 34
Repositorios privados ... 36
Conceptos sobre los repositorios ... 36
Creación de un repositorio para almacenar imágenes ... 37
Pasos a seguir a continuación ... 39
Visualización de detalles de repositorios ... 39
Eliminación de un repositorio ... 40
Políticas de repositorios ... 41
Políticas de repositorios frente a políticas de IAM ... 41
Ejemplos de políticas de repositorio ... 43
Configuración de una instrucción de política de repositorio ... 48
Etiquetado de un repositorio ... 50
Conceptos básicos de etiquetas ... 50
Etiquetado de los recursos para facturación ... 51
Agregar etiquetas. ... 51
Eliminación de etiquetas ... 53
Imágenes privadas ... 55
Insertar una imagen ... 56
Permisos de IAM necesarios ... 56
Inserción de una imagen de Docker ... 58
Empujar una imagen multiarquitectura ... 59
Inserción de un gráfico de Helm ... 61
Firma de una imagen ... 64
Consideraciones ... 64
Requisitos previos ... 64
Configurar la autenticación para el cliente Notary ... 64
Firma de una imagen ... 65
Pasos a seguir a continuación ... 66
Eliminación de artefactos ... 66
Visualización de los detalles de imagen ... 69
Extraer una imagen ... 70
Extracción de la imagen de contenedor Linux de Amazon ... 71
Eliminar una imagen ... 73
Volver a etiquetar una imagen ... 74
Cómo impedir que se sobrescriban las etiquetas de imagen ... 77

Establecer la mutabilidad de las etiquetas de imagen (AWS Management Console) 77
Establecer la mutabilidad de las etiquetas de imagen (AWS CLI) 79
Formatos del manifiesto de imágenes de contenedor 80
Conversión del manifiesto de imagen de Amazon ECR 81
Uso de imágenes de Amazon ECR con Amazon ECS 82
Permisos de IAM necesarios 82
Especificación de una imagen de Amazon ECR en una definición de tarea 84
Uso de imágenes de Amazon ECR con Amazon EKS 85
Permisos de IAM necesarios 85
Instalación de un gráfico de Helm en un clúster de Amazon EKS 86
Escaneo de imágenes para detectar vulnerabilidades 88
Filtros para repositorios 89
Filtro de comodines 89
Escaneo mejorado 90
Consideraciones del escaneo mejorado 90
Cambio de la duración del análisis mejorado 92
Permisos de IAM necesarios 92
Configuración de escaneos mejorados 93
EventBridge eventos 95
Recuperación de resultados 101
Escaneo básico 102
Soporte del sistema operativo para el escaneo básico y el escaneo básico mejorado 103
Configuración de escaneos básicos 106
Cambio al escaneo básico mejorado 106
Escaneo manual de una imagen 108
Recuperación de resultados 109
Solución de problemas de escaneo de imágenes 111
Descripción del estado de análisis SCAN_ELIGIBILITY_EXPIRED 112
Sincronización de un registro principal 113
Plantillas de creación de repositorios 114
Consideraciones para utilizar las reglas de caché de extracción 114
Permisos de IAM necesarios 116
Uso de permisos de registro 117
Pasos a seguir a continuación 119
Configuración de los permisos de ECR multicuenta a ECR PTC 119

Se requieren políticas de IAM para transferir la memoria caché de ECR a ECR entre cuentas ... 119
Creación de una regla de caché de extracción ... 121
Requisitos previos ... 122
Usando el AWS Management Console ... 122
Usando el AWS CLI ... 131
Pasos a seguir a continuación ... 134
Validación de una regla de caché de extracción ... 134
Extracción de una imagen con una regla de caché de extracción ... 136
Almacenar en secreto las credenciales del repositorio principal ... 137
Personalización de los prefijos del repositorio ... 145
Solución de problemas con la caché de extracción ... 146
Replicar imágenes ... 149
Consideraciones sobre la replicación de imágenes privadas ... 149
Ejemplos de replicación ... 151
Ejemplo: Configurar la replicación entre regiones en una única región de destino ... 151
Ejemplo: Configurar la replicación entre regiones mediante un filtro de repositorio ... 151
Ejemplo: Configurar la replicación entre regiones en varias regiones de destino ... 152
Ejemplo: Configurar la replicación entre cuentas ... 152
Ejemplo: Especificar varias reglas en una configuración ... 153
Configuración de la replicación ... 154
Plantillas de creación de repositorios ... 157
Funcionamiento ... 157
Crear una plantilla de creación de repositorios. ... 161
Permisos de IAM para crear plantillas de creación de repositorio ... 161
Crear una política personalizada ... 162
Creación de un rol de IAM ... 163
Cree una plantilla de creación de repositorios ... 164
Actualizar plantillas de creación de repositorios ... 169
Eliminar una plantilla de creación de repositorios ... 170
Automatice la limpieza de imágenes ... 172
Funcionamiento de las políticas de ciclo de vida ... 172
Reglas de evaluación de políticas de ciclo de vida ... 173
Creación de una vista previa de política de ciclo de vida ... 174
Crear una política de ciclo de vida ... 177
Requisito previo ... 177

Ejemplos de políticas de ciclo de vida 179
Plantilla de política de ciclo de vida 179
Filtrar por antigüedad de las imágenes 180
Filtrar por el número de imágenes 180
Filtrar por varias reglas 181
Filtrar por varias etiquetas en una sola regla 184
Filtrar por todas las imágenes 186
Propiedades de la política de ciclo de vida 189
Prioridad de las reglas 189
Descripción 189
Estado de etiqueta 189
Lista de patrones de etiquetas 190
Lista de prefijos de etiqueta 190
Tipo de recuento 191
Unidad de recuento 191
Cantidad 192
Acción 192
Seguridad 193
Identity and Access Management 194
Público 194
Autenticación con identidades 195
Administración de acceso mediante políticas 198
Cómo funciona Amazon Elastic Container Registry con IAM 201
Ejemplos de políticas basadas en identidades 206
Uso del control de acceso basado en etiquetas 211
AWS políticas gestionadas para Amazon ECR 213
Cómo utilizar roles vinculados a servicios 224
Solución de problemas 232
Protección de los datos 234
Cifrado en reposo 236
Validación de conformidad 243
Seguridad de infraestructuras 245
Puntos de conexión de VPC de interfaz (AWS PrivateLink) 245
Prevención de la sustitución confusa entre servicios 254
Monitorización 257
Visualización de Service Quotas y configuración de alarmas 258

Métricas de uso ... 259
Informes de uso de ... 261
Métricas de repositorios ... 261
Habilitar CloudWatch las métricas ... 261
Métricas y dimensiones disponibles ... 262
Visualización de las métricas con CloudWatch ... 262
Eventos y EventBridge ... 263
Eventos de ejemplo de Amazon ECR ... 263
... 267
Registro de acciones de AWS CloudTrail con ... 268
Información de Amazon ECR en CloudTrail ... 268
Descripción de las entradas de archivos de registros de Amazon ECR ... 270
Trabajando con AWS SDKs ... 283
Ejemplos de código ... 285
Conceptos básicos ... 290
Introducción a Amazon ECR ... 290
Conceptos básicos ... 295
Acciones ... 351
Service Quotas ... 403
Administración de las cuotas de servicio de Amazon ECR en la AWS Management Console ... 409
Crear una CloudWatch alarma para monitorear las métricas de uso de la API ... 410
Solución de problemas ... 411
Solución de problemas de Docker ... 411
Los registros de Docker no contienen mensajes de error esperados ... 411
Error «Filesystem Verification Failed» (Fallo al verificar el sistema de archivos) o «404: Image Not Found» (Imagen no encontrada) al extraer una imagen de un repositorio de Amazon ECR ... 412
Error «Filesystem Layer Verification Failed» (Fallo al verificar la capa del sistema de archivos) al extraer imágenes de Amazon ECR ... 413
Errores HTTP 403 o error «no basic auth credentials» (no hay credenciales de autenticación básica) al insertar contenido en un repositorio ... 413
Solución de problemas de los mensajes de error de Amazon ECR ... 414
HTTP 429: demasiadas solicitudes o ThrottleException ... 414
HTTP 403: "User [arn] is not authorized to perform [operación]" ... 415
Error HTTP 404: «Repository Does Not Exist» (El repositorio no existe) ... 415

Error: No se puede realizar un inicio de sesión interactivo desde un dispositivo que no sea TTY 416
Uso de Podman con Amazon ECR 417
Uso de Podman para la autenticación con Amazon ECR 417
Uso del asistente de credenciales Amazon ECR con Podman 417
Extracción de imágenes desde Amazon ECR mediante Podman 417
Ejecución de contenedores para Amazon ECR con Podman 418
Inserción de imágenes en Amazon ECR con Podman 418
Historial de documentos 419
..... cdxxviii

¿Qué es Amazon Elastic Container Registry?

Amazon Elastic Container Registry (Amazon ECR) es AWS un servicio gestionado de registro de imágenes de contenedores seguro, escalable y fiable. Amazon ECR admite repositorios privados con permisos basados en recursos mediante IAM. AWS Esto es para que los usuarios o EC2 instancias de Amazon específicos puedan acceder a tus repositorios e imágenes de contenedores. Puede utilizar la CLI preferida para insertar, extraer y administrar imágenes de Docker, imágenes de Open Container Initiative (OCI) y artefactos compatibles con OCI.

> **Note**
>
> Amazon ECR también admite repositorios de imágenes de contenedor públicos. Para obtener más información, consulte ¿Qué es Amazon ECR Public? en la Guía del usuario de Amazon ECR Public.

El equipo de servicios de AWS contenedores mantiene una hoja de ruta pública. GitHub Contiene información sobre en qué están trabajando los equipos y permite a todos AWS los clientes dar su opinión directamente. Para obtener más información, consulte Guía de contenedores de AWS.

Conceptos y componentes de Amazon ECR

Amazon ECR es un servicio de registro de contenedor de Docker completamente administrado que es proporcionado por AWS. Le permite almacenar, administrar e implementar imágenes de contenedores de Docker de forma segura y fiable. Estos conceptos y componentes funcionan en conjunto para proporcionar un servicio de registro de contenedores Docker seguro, escalable y fiable dentro del sistema AWS, que le permite gestionar e implementar de forma eficiente sus aplicaciones contenerizadas.

Estos son algunos conceptos y componentes clave de Amazon ECR:

Registro

Un registro de Amazon ECR es un repositorio privado que se proporciona a cada AWS cuenta, donde puede crear uno o más repositorios. Estos repositorios le permiten almacenar y distribuir imágenes de Docker, imágenes de Open Container Initiative (OCI) y otros artefactos compatibles con OCI en su entorno. AWS Para obtener más información, consulte Registro privado de Amazon ECR.

Token de autorización

Su cliente debe autenticarse en un registro privado de Amazon ECR como usuario de AWS antes de poder enviar y extraer imágenes. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

Repositorio

Un repositorio de Amazon ECR es una colección lógica donde puede almacenar las imágenes de Docker, imágenes de Iniciativa de contenedores abiertos (OCI) y otros artefactos compatibles con OCI. Dentro de un registro de Amazon ECR, puede tener varios repositorios para organizar las imágenes de sus contenedores. Para obtener más información, consulte Repositorios privados de Amazon ECR.

Política sobre repositorios

Puede controlar el acceso a los repositorios y al contenido que portan mediante políticas. Para obtener más información, consulte Políticas de repositorios privados en Amazon ECR.

Imagen

Puede insertar y extraer imágenes de contenedor en los repositorios y utilizarlas localmente en su sistema de desarrollo o en definiciones de tareas de Amazon ECS y especificaciones del pod de Amazon EKS. Para obtener más información, consulte Uso de imágenes de Amazon ECR con Amazon ECS y Uso de imágenes de Amazon ECR con Amazon EKS.

Política del ciclo de vida

Las políticas de ciclo de vida de Amazon ECR le permiten gestionar el ciclo de vida de sus imágenes mediante la definición de reglas para eliminar y caducar las imágenes antiguas o no utilizadas. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.

Escaneo de imágenes

Amazon ECR proporciona un análisis de imágenes integrado para ayudar a identificar vulnerabilidades de software en las imágenes de contenedor. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.

Control de acceso

Amazon ECR utiliza IAM para controlar el acceso a sus repositorios. Puede crear usuarios, grupos y roles de IAM con permisos específicos para insertar, extraer o gestionar los repositorios

de Amazon ECR. Para obtener más información, consulte Seguridad en Amazon Elastic Container Registry.

Replicación entre regiones y entre cuentas

Amazon ECR admite la replicación de imágenes en varias AWS cuentas y regiones para aumentar la disponibilidad y reducir la latencia. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR.

Cifrado

Amazon ECR admite el cifrado del lado del servidor de las imágenes de Docker en reposo mediante AWS KMS. Para obtener más información, consulte Protección de los datos en Amazon ECR.

AWS Command Line Interface Integration

AWS CLI Proporciona comandos para interactuar con los repositorios de Amazon ECR, como crear, enumerar, insertar y extraer imágenes.

AWS Management Console

Amazon ECR también se puede administrar a través de AWS Management Console, que proporciona una interfaz web fácil de usar para trabajar con sus repositorios e imágenes.

AWS CloudTrail

Amazon ECR se integra con Amazon ECR AWS CloudTrail, lo que le permite registrar y auditar las llamadas a la API realizadas a Amazon ECR por motivos de seguridad y conformidad. Para obtener más información, consulte Registrar las acciones de Amazon ECR con AWS CloudTrail.

Amazon CloudWatch

Amazon ECR proporciona métricas y registros que se pueden monitorear mediante Amazon CloudWatch, lo que le permite realizar un seguimiento del rendimiento y el uso de sus repositorios de Amazon ECR. Para obtener más información, consulte Métricas de repositorios de Amazon ECR.

Casos de uso comunes en Amazon ECR

Amazon ECR es un servicio de registro de contenedor de Docker completamente administrado que ofrece AWS. Proporciona un repositorio seguro y escalable para almacenar y distribuir imágenes de contenedores de Docker, lo que lo convierte en un componente esencial para la implementación

de aplicaciones en contenedores. Amazon ECR simplifica el proceso de creación, distribución y ejecución de aplicaciones en contenedores en varios AWS servicios y entornos locales.

A continuación, se indican algunos casos de uso clave de Amazon ECR:

Almacenamiento y distribución de imágenes de contenedores

Amazon ECR actúa como un repositorio centralizado para almacenar y distribuir imágenes de contenedores de Docker dentro de una organización o para el consumo público. Los desarrolladores pueden enviar las imágenes de sus contenedores a Amazon ECR y, a continuación, extraerlas de cualquier entorno informático interno AWS EC2 AWS Fargate, como Amazon o Amazon EKS. Para obtener más información, consulte Repositorios privados de Amazon ECR.

Implementación e integración continuas (CI/CD)

Amazon ECR se integra a la perfección con AWS CodeBuild y otras CI/CD tools, enabling automated building, testing, and deployment of containerized applications. Container images can be automatically pushed to Amazon ECR as part of the CI/CD canalizaciones, lo que garantiza una implementación uniforme y confiable en diferentes entornos. AWS CodePipeline

Arquitectura de microservicios

Amazon ECR es ideal para arquitecturas de microservicios, en las que las aplicaciones se dividen en servicios disociados más pequeños empaquetados como contenedores. Cada microservicio puede tener su propia imagen de contenedor almacenada en Amazon ECR, lo que permite el desarrollo, la implementación y el escalado independientes de los servicios individuales.

Implementaciones híbridas y multinube

Amazon ECR admite la posibilidad de extraer imágenes de contenedores desde otros registros de contenedores, como Docker Hub o registros de terceros. Esto permite a las organizaciones mantener un modelo de implementación coherente en entornos híbridos o multinube mediante el uso de Amazon ECR como repositorio central de imágenes de contenedores.

Control de acceso y seguridad

Amazon ECR proporciona mecanismos de control de acceso detallados que permiten a las organizaciones controlar quién puede insertar imágenes de contenedores en el registro o extraerlas. También se integra AWS Identity and Access Management para la autenticación y la autorización, lo que garantiza un acceso seguro a las imágenes del contenedor. Para obtener más información, consulte Seguridad en Amazon Elastic Container Registry.

Análisis de vulnerabilidades de imágenes

Amazon ECR ofrece un escaneo automático de las imágenes de los contenedores para detectar vulnerabilidades de software y posibles errores de configuración, lo que ayuda a mantener un entorno de contenedores seguro y compatible. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.

Registro privado de contenedores

Para las organizaciones con requisitos estrictos de seguridad o conformidad, Amazon ECR se puede utilizar como un registro de contenedores privado, lo que garantiza que las imágenes confidenciales de los contenedores no se expongan a los registros públicos y solo se pueda acceder a ellas dentro del AWS entorno de la organización. Para obtener más información, consulte Registro privado de Amazon ECR.

Implementación de aplicaciones distribuidas globalmente con la replicación de Amazon ECR

Al aprovechar la capacidad de replicación de Amazon ECR, puede centralizar las imágenes de aplicaciones web en contenedores en un repositorio principal, lo que permite la distribución automática en varias AWS regiones, garantiza despliegues globales consistentes con baja latencia en todo el mundo y reduce la carga operativa. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR

Limpieza automática de imágenes de contenedores obsoletas

Las políticas de ciclo de vida de Amazon ECR permiten la limpieza automática de imágenes de contenedores obsoletas en función de reglas definidas, como la antigüedad, el número o las etiquetas, lo que optimiza los costos de almacenamiento, mantiene un registro organizado, mejora la seguridad y el cumplimiento y optimiza los flujos de trabajo de desarrollo mediante la automatización. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR

Características de Amazon ECR

Amazon ECR ofrece las siguientes características:

- Las políticas de ciclo de vida ayudan a administrar el ciclo de vida de las imágenes en sus repositorios. Debe definir reglas que den como resultado la limpieza de imágenes no utilizadas. Puede probar reglas antes de aplicarlas al repositorio. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.

- El escaneo de imágenes permite identificar vulnerabilidades de software en las imágenes de contenedor. Los repositorios se pueden configurar para escanear al insertar. Esto garantiza que cada nueva imagen insertada al repositorio se escanee. A continuación, puede recuperar los resultados del escaneo de la imagen. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.
- La replicación entre regiones y entre cuentas hace que sea más fácil tener las imágenes donde las necesite. Se configura como un parámetro del Registro y se aplica por región. Para obtener más información, consulte Configuración del registro privado en Amazon ECR.
- Las reglas de caché de extracción proporcionan una manera de almacenar en caché los repositorios en un registro ascendente en su registro privado de Amazon ECR. Al utilizar una regla de caché de extracción, Amazon ECR se comunicará periódicamente con el registro ascendente para garantizar que la imagen almacenada en caché en su registro privado de Amazon ECR esté actualizada. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.

Primeros pasos con Amazon ECR

Si utiliza Amazon Elastic Container Service (Amazon ECS) o Amazon Elastic Kubernetes Service (Amazon EKS), tenga en cuenta que la configuración de estos dos servicios es similar a la configuración de Amazon ECR, ya que Amazon ECR es una extensión de ambos.

Cuando utilice Amazon ECR, utilice una versión de AWS CLI que sea compatible AWS Command Line Interface con las funciones más recientes de Amazon ECR. Si no ve compatibilidad con una función de Amazon ECR en AWS CLI, actualice a la última versión de. AWS CLI Para obtener información sobre la instalación de la última versión de AWS CLI, consulte Instalar o actualizar a la última versión de AWS CLI en la Guía del AWS Command Line Interface usuario.

Para obtener más información acerca de cómo insertar una imagen de contenedor en un repositorio privado de Amazon ECR mediante el uso de AWS CLI y Docker, consulte Desplazamiento de una imagen a lo largo de su ciclo de vida en Amazon ECR.

Precios de Amazon ECR

Con Amazon ECR, solo pagará por la cantidad de datos que almacene en sus repositorios, y por la transferencia de datos de sus inserciones y extracciones de imágenes. Para obtener más información, consulte los Precios de Amazon ECR.

Desplazamiento de una imagen a lo largo de su ciclo de vida en Amazon ECR

Si utiliza Amazon ECR por primera vez, siga los siguientes pasos con la CLI de Docker y AWS CLI para crear una imagen de muestra, autenticarse en el registro predeterminado y crear un repositorio privado. A continuación, inserte una imagen en el repositorio privado y extráigala. Cuando termine, borre la imagen de muestra y el repositorio.

Para usar el AWS Management Console en lugar del, consulte. AWS CLIthe section called “Creación de un repositorio para almacenar imágenes”

Para obtener más información sobre las demás herramientas disponibles para administrar AWS los recursos, incluidos los distintos AWS SDKs kits de herramientas del IDE y las herramientas de línea de PowerShell comandos de Windows, consulte http://aws.amazon.com/tools/.

Requisitos previos

Si no dispone de la última versión AWS CLI ni de Docker instalado y listo para usarse, siga estos pasos para instalar ambas herramientas.

Instale el AWS CLI

Para usarlo AWS CLI con Amazon ECR, instale la AWS CLI versión más reciente. Para obtener información, consulte Instalar la AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

Instalar Docker

Docker está disponible en muchos sistemas operativos diferentes, incluidas las distribuciones de Linux más modernas, como Ubuntu, e incluso en macOS y Windows. Para obtener más información sobre cómo instalar Docker en su sistema operativo concreto, consulte la guía de instalación de Docker.

No necesita un sistema de desarrollo local para usar Docker. Si EC2 ya utilizas Amazon, puedes lanzar una instancia de Amazon Linux 2023 e instalar Docker para empezar.

Si ya tiene Docker instalado, pase a Paso 1: creación de una imagen de Docker.

Para instalar Docker en una EC2 instancia de Amazon mediante una AMI de Amazon Linux 2023

1. Lance una instancia con la AMI de Amazon Linux 2023. Para obtener más información, consulta Cómo lanzar una instancia en la Guía del EC2 usuario de Amazon.
2. Conecte con la instancia . Para obtener más información, consulte Connect to Your Linux Instance en la Guía del EC2 usuario de Amazon.
3. Actualice la caché de paquetes y los paquetes instalados en la instancia.

```
sudo yum update -y
```

4. Instale el paquete de Community Edition de Docker más reciente.

```
sudo yum install docker
```

5. Abra el servicio de Docker.

```
sudo service docker start
```

6. Agregue el `ec2-user` al grupo `docker` para que pueda ejecutar comandos de Docker sin usar `sudo`.

```
sudo usermod -a -G docker ec2-user
```

7. Cierre sesión y vuelva a iniciarla para actualizar los nuevos permisos de grupo de `docker`. Para ello, cierre la ventana de su terminal de SSH actual y vuelva a conectarse a la instancia en una ventana nueva. De esta forma, la nueva sesión de SSH tendrá los permisos de grupo de `docker` adecuados.
8. Compruebe que el `ec2-user` puede ejecutar comandos de Docker sin `sudo`.

```
docker info
```

> **Note**
>
> En algunos casos, es posible que tenga que reiniciar su instancia para que el `ec2-user` tenga los permisos necesarios para acceder al daemon de Docker. Intente reiniciar su instancia si ve el siguiente error:

```
Cannot connect to the Docker daemon. Is the docker daemon running on this
 host?
```

Paso 1: creación de una imagen de Docker

En este paso, crearás una imagen de Docker de una aplicación web sencilla y la probarás en tu sistema local o en una EC2 instancia de Amazon.

Creación de una imagen Docker de una aplicación web simple

1. Cree un archivo denominado `Dockerfile`. Un Dockerfile es un manifiesto que describe la imagen base para su imagen Docker y qué desea instalar y que se ejecute en ella. Para obtener más información acerca de los archivos Docker, consulte Docker Reference.

   ```
   touch Dockerfile
   ```

2. Edite el `Dockerfile` que acaba de crear y agregue el siguiente contenido.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest

   # Install dependencies
   RUN yum update -y && \
    yum install -y httpd

   # Install apache and write hello world message
   RUN echo 'Hello World!' > /var/www/html/index.html

   # Configure apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
    echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh

   EXPOSE 80

   CMD /root/run_apache.sh
   ```

Este Dockerfile utiliza la imagen pública de Amazon Linux 2 alojada en Amazon ECR Public. Las instrucciones RUN actualizan la caché del paquete, instalan algunos paquetes de software para el servidor web y, a continuación, escriben el contenido “Hello World!”. en la raíz de documentos del servidor web. El folleto EXPOSE expone el puerto 80 en el contenedor y las instrucciones CMD inician el servidor web.

3. Cree la imagen Docker desde el Dockerfile.

> **Note**
>
> Algunas versiones de Docker pueden requerir la ruta completa a su Dockerfile en el siguiente comando en lugar de la ruta relativa que se muestra a continuación.

```
docker build -t hello-world .
```

4. Incluya la imagen de su contenedor.

```
docker images --filter reference=hello-world
```

Salida:

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. Ejecute la nueva imagen. La opción `-p 80:80` asigna el puerto 80 expuesto en el contenedor al puerto 80 del sistema de host. Para obtener más información acerca de docker run, diríjase a la referencia de ejecución de Docker.

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> La salida desde el servidor web Apache se muestra en la ventana de la terminal. Puede hacer caso omiso del mensaje "`Could not reliably determine the fully qualified domain name`"

6. Abra un navegador y encuentre el servidor que está ejecutando Docker y alojando su contenedor.

   - Si utilizas una EC2 instancia, este es el valor de DNS público del servidor, que es la misma dirección que utilizas para conectarte a la instancia mediante SSH. Asegúrese de que el grupo de seguridad para la instancia permite el tráfico entrante en el puerto 80.
   - Si ejecuta Docker de forma local, dirija el navegador a http://localhost/.
   - Si lo utiliza docker-machine en un ordenador Windows o Mac, busque la dirección IP de la VirtualBox máquina virtual que aloja Docker con el docker-machine ip comando y sustitúyala por el *`machine-name`* nombre de la máquina docker que está utilizando.

     ```
     docker-machine ip machine-name
     ```

   Debería ver una página web que diga "Hello, World!" statement.

7. Detenga el contenedor de Docker escribiendo Ctrl + c.

Paso 2: crear un repositorio

Ahora que ya tiene una imagen para insertar en Amazon ECR, debe crear un repositorio para almacenarla. En este ejemplo, va a crear un repositorio llamado `hello-repository` en el que insertará después la imagen `hello-world:latest`. Para crearlo, ejecute este comando:

```
aws ecr create-repository \
    --repository-name hello-repository \
    --region region
```

Paso 3: autenticar en su registro predeterminado

Una vez que haya instalado y configurado AWS CLI, autentique la CLI de Docker en su registro predeterminado. De este modo, el comando docker puede extraer e insertar imágenes con Amazon ECR. AWS CLI Proporciona un get-login-password comando para simplificar el proceso de autenticación.

Para autenticar Docker en un registro de Amazon ECR con get-login-password, ejecute el comando. aws ecr get-login-password Al pasar el token de autenticación al comando docker login, utilice el valor AWS para el nombre de usuario y especifique el URI del registro de Amazon ECR en el que desea autenticarse. Si se autentica en varios registros, deberá repetir el comando con cada registro.

> **Important**
>
> Si recibe un error, instale o actualice a la versión más reciente de la AWS CLI. Para obtener más información, consulte Installing the AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

- get-login-password (AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- Get- Command () ECRLoginAWS Tools for Windows PowerShell

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Paso 4: Insertar una imagen en Amazon ECR

Ahora puede insertar la imagen en el repositorio de Amazon ECR que ha creado en la sección anterior. Utilice la CLI de docker para insertar imágenes cuando se cumplan los siguientes requisitos previos:

- La versión instalada es la versión mínima de docker: 1.7.
- El token de autorización de Amazon ECR se configuró con docker login.

- Debe existir el repositorio de Amazon ECR y el usuario debe disponer de acceso para insertar imágenes en él.

Si se cumplen estos requisitos previos, podrá insertar la imagen en el repositorio recién creado del registro predeterminado de su cuenta.

Etiquetado e inserción de una imagen en Amazon ECR

1. Muestre las imágenes que ha almacenado localmente para identificar la que desea etiquetar e insertar.

   ```
   docker images
   ```

   Salida:

   ```
   REPOSITORY          TAG                 IMAGE ID            CREATED
    VIRTUAL SIZE
   hello-world         latest              e9ffedc8c286        4 minutes ago
    241MB
   ```

2. Etiquete la imagen que desea insertar en el repositorio.

   ```
   docker tag hello-world:latest aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository
   ```

3. Inserte la imagen.

   ```
   docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
   ```

   Salida:

   ```
   The push refers to a repository [aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository] (len: 1)
   e9ae3c220b23: Pushed
   a6785352b25c: Pushed
   0998bf8fb9e9: Pushed
   0a85502c06c9: Pushed
   latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
    size: 6774
   ```

Paso 5: Extraer una imagen de Amazon ECR

Una vez que se inserta la imagen en el repositorio de Amazon ECR, puede extraerla de otras ubicaciones. Utilice la CLI de docker para extraer imágenes cuando se cumplan los siguientes requisitos previos:

- La versión instalada es la versión mínima de docker: 1.7.
- El token de autorización de Amazon ECR se configuró con docker login.
- Debe existir el repositorio de Amazon ECR y el usuario debe tener acceso para extraer imágenes de él.

Si se cumplen estos requisitos previos, podrá extraer la imagen. Para extraer la imagen de ejemplo de Amazon ECR, ejecute este comando:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
```

Salida:

```
latest: Pulling from hello-repository
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
Status: Downloaded newer image for aws_account_id.dkr.region.amazonaws.com/hello-
repository:latest
```

Paso 6: eliminar una imagen

Si ya no necesita una imagen de uno de los repositorios, puede eliminarla. Para eliminar una imagen, especifique el repositorio en el que esta se encuentra y seleccione un valor `imageTag` o `imageDigest`. En el ejemplo siguiente, se elimina una imagen del repositorio `hello-repository` con la etiqueta de imagen `latest`. Para eliminar la imagen de ejemplo del repositorio, ejecute el siguiente comando:

```
aws ecr batch-delete-image \
      --repository-name hello-repository \
```

```
      --image-ids imageTag=latest \
      --region region
```

Paso 7: eliminar un repositorio

Si ya no necesita un repositorio completo de imágenes, puede eliminarlo. En el siguiente ejemplo, se utiliza la marca `--force` para eliminar un repositorio que contiene imágenes. Para eliminar un repositorio que contiene imágenes (y todas las imágenes con él), ejecute este comando:

```
aws ecr delete-repository \
      --repository-name hello-repository \
      --force \
      --region region
```

Optimización del rendimiento en Amazon ECR

Puede utilizar la siguiente información sobre las configuraciones y estrategias para optimizar el rendimiento al utilizar Amazon ECR.

Usar Docker 1.10 o versiones posteriores para poder realizar cargas en capas simultáneas

Las imágenes de Docker se componen de capas, que son etapas intermedias de compilación de la imagen. Cada línea de un archivo Dockerfile genera una nueva capa. Al utilizar Docker 1.10 o versiones posteriores, Docker envía de forma predeterminada tantas capas como cargas simultáneas sea posible enviar a Amazon ECR, lo que da como resultado tiempos de carga más rápidos.

Usar una imagen base más pequeña

Las imágenes predeterminadas disponibles en Docker Hub pueden contener varias dependencias que no son necesarias para la aplicación. Plantéese utilizar una imagen más pequeña creada y mantenida por otros miembros de la comunidad Docker o cree su propia imagen base a partir de la imagen de prueba mínima de Docker. Para obtener más información, consulte Create a base image en la documentación de Docker.

Sitúe las dependencias que menos cambian en la primera parte del archivo Dockerfile

Docker almacena capas en caché, lo que incrementa la velocidad de compilación. Si una capa no ha cambiado desde la última compilación, Docker recurre a la versión en caché en lugar de compilarla de nuevo. Sin embargo, cada capa depende de las que le anteceden. Si una capa cambia, Docker compila de nuevo no solo esa capa, sino también las que le siguen.

Para reducir al mínimo el tiempo necesario para compilar de nuevo un archivo Docker y volver a cargar las capas, plantéese situar las dependencias que cambian con menos frecuencia en el archivo Docker. Sitúe las dependencias que cambian rápidamente (como, por ejemplo, el código fuente de la aplicación) más adelante en la pila.

Encadene los comandos para evitar almacenar archivos innecesarios.

Los archivos intermedios que se crean en una capa siguen formando parte de ella aunque se eliminen en una capa posterior. Considere el siguiente ejemplo:

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz
RUN wget tar -xvf software.tar.gz
```

```
RUN mv software/binary /opt/bin/myapp
RUN rm software.tar.gz
```

En este ejemplo, las capas creadas por el primer y el segundo comando RUN contienen el archivo original .tar.gz y su contenido descomprimido. Esto ocurre a pesar de que el cuarto comando RUN elimina el archivo .tar.gz. Estos comandos se pueden encadenar en una única instrucción RUN para asegurarse de que los archivos innecesarios no formen parte de la imagen final de Docker:

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz &&\
    wget tar -xvf software.tar.gz &&\
    mv software/binary /opt/bin/myapp &&\
    rm software.tar.gz
```

Usar el punto de enlace regional más cercano

Para reducir la latencia al extraer imágenes de Amazon ECR, utilice el punto de enlace regional más cercano a la zona de ejecución de la aplicación. Si tu aplicación se ejecuta en una EC2 instancia de Amazon, puedes usar el siguiente código de shell para obtener la región de la zona de disponibilidad de la instancia:

```
REGION=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone
 |\
     sed -n 's/\(\d*\)[a-zA-Z]*$/\1/p')
```

La región se puede pasar a AWS CLI los comandos mediante el --region parámetro, o se puede establecer como la región predeterminada para un perfil mediante el aws configure comando. También puedes configurar la región al realizar llamadas con el AWS SDK. Para obtener más información, consulte la documentación del SDK de su lenguaje de programación específico.

Realizar solicitudes a los registros de Amazon ECR

Puede insertar, extraer, eliminar, ver y administrar imágenes de OCI, imágenes de Docker y artefactos compatibles con OCI en los registros privados de Amazon ECR mediante puntos de enlace IPv4 exclusivos o puntos de enlace de doble pila (y). IPv4 IPv6 Para realizar solicitudes desde redes, puede utilizar una pila doble o puntos de enlace IPv4 . IPv4 Para realizar solicitudes desde una IPv6 red, utilice un punto final de doble pila. Para obtener más información sobre cómo realizar solicitudes a los registros públicos de Amazon ECR mediante IPv4 puntos de enlace de doble pila, consulte Realizar solicitudes a registros públicos de Amazon ECR. El acceso a Amazon ECR a través IPv6 de Amazon ECR no conlleva cargos adicionales. Para obtener más información sobre los precios, consulte los precios de Amazon Elastic Container Registry.

> **Note**
>
> Amazon ECR no admite el AWS PrivateLink tráfico a través de puntos de enlace de doble pila. Si necesita asistencia, debe utilizar IPv4 únicamente puntos de enlace de Amazon ECR. AWS PrivateLink

Los puntos de enlace de Amazon ECR se designan mediante atributos que van más allá de la compatibilidad con puntos IPv4 de enlace exclusivos o de doble pila. Estos atributos pueden incluir:

- Región: cada punto final es específico de una región.
- Tipo: la selección del punto final depende de si utiliza el AWS SDK o las interfaces de línea de comandos de Docker o compatibles con OCI.
- Seguridad: en determinadas regiones, Amazon ECR ofrece puntos de enlace compatibles con FIPS. Para obtener más información sobre una lista de puntos de enlace Amazon ECR que cumplen con FIPS, consulte la Norma Federal de Procesamiento de Información (FIPS) 140-3.

Para obtener más información sobre los puntos de enlace de servicio compatibles IPv4 con el cliente de doble pila, Docker y OCI, que gestiona las llamadas a la API de Amazon ECR desde la AWS CLI, y AWS SDKs consulte Puntos de enlace de servicio.

Cómo empezar a realizar solicitudes IPv6

Para realizar una solicitud a un registro de Amazon ECR IPv6, debe utilizar un punto de conexión de doble pila. Antes de acceder a un registro de Amazon ECR IPv6, compruebe los siguientes requisitos:

- Su cliente y su red deben ser compatibles IPv6.
- Amazon ECR admite los siguientes tipos de solicitudes: IPv6
  - Solicitudes de clientes de OCI y Docker:

    `<registry-id>.dkr-ecr.<aws-region>.on.aws`
  - AWS Solicitudes de API:

    `ecr.<aws-region>.api.aws`
- Debe actualizar todas las políticas AWS Identity and Access Management (de IAM) o de registro que utilicen el filtrado de direcciones IP de origen para incluir los intervalos de IPv6 direcciones. Para obtener más información, consulte Uso de IPv6 direcciones en las políticas de IAM.
- Cuando lo usa IPv6, los registros de acceso al servidor muestran `Remote IP` las direcciones en IPv6 formato. Actualice las herramientas, los scripts y el software existentes para analizar estas direcciones IP IPv6 con formato.

> **Note**
>
> Si tiene problemas relacionados con la presencia de IPv6 direcciones en los archivos de registro, póngase en contacto con nosotros. AWS Support

Probar la compatibilidad de dirección IP

Si utiliza Linux/Unix o Mac OS X, puede comprobar si puede acceder a un punto final de doble pila IPv6 mediante el `curl` comando que se muestra en el siguiente ejemplo:

Example

```
curl --verbose https://ecr.us-west-2.api.aws
```

Usted obtiene información similar a la del siguiente ejemplo. Si está conectado a través de IPv6 la dirección IP conectada, será una dirección. IPv6

```
* About to connect() to ecr.us-west-2.api.aws port 443 (#0)
* Trying IPv6 address... connected
* Connected to ecr.us-west-2.api.aws (IPv6 address) port 443 (#0)
> Host: ecr.us-west-2.api.aws
* Request completely sent off
```

Si utiliza Microsoft Windows 7 o Windows 10, puede probar si puede acceder a un punto final de doble pila a través IPv4 o IPv6 mediante el `ping` comando que se muestra en el siguiente ejemplo.

```
ping ecr.us-west-2.api.aws
```

Realización de solicitudes IPv6 mediante puntos de enlace de doble pila

Puede realizar llamadas a la API Amazon ECR IPv6 mediante puntos de enlace de doble pila. La funcionalidad y el rendimiento de las operaciones de la API de Amazon ECR se mantienen uniformes independientemente de si utiliza IPv4 o IPv6.

Cuando usa AWS Command Line Interface (AWS CLI) y AWS SDKs, puede habilitarlo IPv6 mediante un parámetro o indicador para cambiar a un punto final de doble pila, o especificando directamente el punto final de doble pila en su archivo de configuración para anular el punto de enlace predeterminado de Amazon ECR. También puede realizar cambios en la configuración mediante un comando, que se establece en true en `use_dualstack_endpoint` el perfil predeterminado. Para obtener más información`use_dualstack_endpoint`, consulte los puntos finales FIPS y de doble pila.

Example Realizar cambios de configuración mediante un comando

```
aws configure set default.ecr.use_dualstack_endpoint true
```

Example Hacer solicitudes en lugar de IPv6 usar AWS CLI

```
aws ecr describe-repositories --region us-west-2 --endpoint-url https://
ecr.us-west-2.api.aws
```

Uso de puntos de enlace de Amazon ECR desde la CLI de docker

Tras iniciar sesión en el repositorio de Amazon ECR y etiquetar la imagen, puede insertar y extraer imágenes de OCI e imágenes de Docker a los registros de Amazon ECR y extraerlos de ellos. Los siguientes ejemplos muestran los comandos docker push y docker pull con ambos puntos finales de doble pila.

Example Empujar imágenes de docker mediante un punto final IPv4

```
docker push <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example Empujar imágenes de docker mediante un punto final de doble pila

```
docker push <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Example Extracción de imágenes de docker mediante un punto final IPv4

```
docker pull <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example Extracción de imágenes de docker mediante un punto final de doble pila

```
docker pull <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Uso de IPv6 direcciones en las políticas de IAM

Antes de acceder a un registro mediante IPv6, asegúrese de que su usuario de IAM y las políticas de registro de Amazon ECR que utilizan el filtrado de direcciones IP incluyan intervalos de IPv6 direcciones. Si las políticas de filtrado de direcciones IP no se actualizan para gestionar IPv6 las direcciones, es posible que los clientes pierdan u obtengan acceso al registro de forma incorrecta al empezar a usarlo. IPv6 Para obtener más información acerca de cómo administrar los permisos de acceso con IAM, consulte Identity and Access Management para Amazon Elastic Container Registry.

Las políticas de IAM que filtran direcciones IP utilizan operadores de condición de dirección IP. El siguiente ejemplo de política de registro muestra cómo identificar el `54.240.143.*` rango de IPv4 direcciones permitidas mediante operadores de condición de direcciones IP. A las direcciones IP que se encuentren fuera de este rango se les deniega el acceso al registro (`exampleregistry`). Como todas IPv6 las direcciones están fuera del rango permitido, esta política impide el acceso de IPv6 las direcciones`exampleregistry`.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "ecr:*",
      "Resource": "arn:aws:ecr:::exampleregistry/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

Para permitir los rangos de direcciones IPv4 (`54.240.143.0/24`) y IPv6 (`2001:DB8:1234:5678::/64`), modifique el elemento Condition de la política de registro como se muestra en el siguiente ejemplo. Puede utilizar este formato de `Condition` bloque para actualizar las políticas de registro y de usuario de IAM.

```
      "Condition": {
        "IpAddress": {
           "aws:SourceIp": [
             "54.240.143.0/24",
              "2001:DB8:1234:5678::/64"
            ]
         }
       }
```

Important

Antes de usarlo IPv6 , debe actualizar todas las políticas de registro y usuario de IAM pertinentes que utilizan el filtrado de direcciones IP. No recomendamos utilizar el filtrado de direcciones IP en las políticas de registro.

Puede revisar sus políticas de usuario de IAM en la consola de IAM en. https://console.aws.amazon.com/iam/ Para obtener más información acerca de IAM, consulte la guía del usuario de IAM.

Registro privado de Amazon ECR

Los registros privados de Amazon ECR alojan sus imágenes de contenedor en una arquitectura escalable y de alta disponibilidad. Puede utilizar el registro privado para administrar repositorios privados de imágenes que consten de imágenes y artefactos de Docker y Open Container Initiative (OCI). Cada cuenta AWS se proporciona con un registro privado de Amazon ECR predeterminado. Para obtener más información acerca de los registros públicos de Amazon ECR, consulte Registros públicos en la Guía del usuario de Amazon Elastic Container Registry.

Conceptos de registro privado

- La URL del registro privado predeterminado es `https://aws_account_id.dkr.ecr.region.amazonaws.com`.
- De forma predeterminada, su cuenta tiene acceso de lectura y escritura en los repositorios que cree en el registro privado. Sin embargo, los usuarios necesitan permisos para realizar llamadas al Amazon ECR APIs y para enviar o extraer imágenes de sus repositorios privados. Amazon ECR proporciona varias políticas administradas por para controlar el acceso de los usuarios en distintos niveles. Para obtener más información, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.
- Para poder usar los comandos docker push y docker pull con objeto de insertar imágenes en los repositorios del registro privado y extraerlas de estos, es necesario autenticar el cliente Docker en ese registro. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.
- Los repositorios privados se pueden controlar mediante políticas tanto de acceso de usuarios de como de repositorio. Para obtener más información acerca de las políticas de repositorios, consulte Políticas de repositorios privados en Amazon ECR.
- Los repositorios de su registro privado se pueden replicar en todas AWS las regiones en su propio registro privado y en cuentas independientes configurando la replicación para su registro privado. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR.

Autenticación de registro privado en Amazon ECR

Puede usar el AWS Management Console, el o el AWS CLI AWS SDKs para crear y administrar repositorios privados. También puede utilizar estos métodos para realizar determinadas acciones con las imágenes, por ejemplo listarlas o eliminarlas. Estos clientes utilizan métodos de AWS

autenticación estándar. Aunque es posible usar la API de Amazon ECR para insertar y extraer imágenes, es más probable que use la CLI de Docker o una biblioteca de Docker específica del lenguaje.

La CLI de Docker no admite métodos de autenticación de IAM nativos. Se deben tomar pasos adicionales para que Amazon ECR pueda autenticar y autorizar solicitudes de inserción y extracción de Docker.

Los métodos de autenticación del registro que se detallan en la sección siguiente están disponibles.

Uso del ayudante de credenciales de Amazon ECR

Amazon ECR proporciona un ayudante de credenciales de Docker que hace que sea más fácil almacenar y usar credenciales de Docker cuando se insertan y se extraen imágenes de Amazon ECR. Para obtener información sobre los pasos de instalación y configuración, consulte Amazon ECR Docker Credential Helper.

> **Note**
>
> El asistente de credenciales de Amazon ECR Docker no admite la autenticación multifactor (MFA).

Uso de un token de autorización

El ámbito de permiso de un token de autorización es igual que el de la entidad de seguridad de IAM que se utiliza para recuperar el token de autenticación. Se utiliza un token de autenticación para acceder a un registro de Amazon ECR al que la entidad de seguridad de IAM tiene acceso y que es válido durante 12 horas. Para obtener un token de autorización, debe usar la operación de GetAuthorizationTokenAPI para recuperar un token de autorización codificado en base64 que contenga el nombre de usuario `AWS` y una contraseña codificada. El AWS CLI `get-login-password` comando lo simplifica al recuperar y decodificar el token de autorización, que luego puedes canalizar a un comando para autenticarlo. docker login

Autenticación de Docker en un registro privado de Amazon ECR con get-login

- Para autenticar Docker en un registro de Amazon ECR con get-login-password, ejecute el comando. aws ecr get-login-password Al pasar el token de autenticación al comando docker login, utilice el valor `AWS` para el nombre de usuario y especifique el URI del registro de Amazon

ECR en el que desea autenticarse. Si se autentica en varios registros, deberá repetir el comando con cada registro.

> **Important**
>
> Si recibe un error, instale o actualice a la versión más reciente de la AWS CLI. Para obtener más información, consulte Installing the AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

- get-login-password (AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- Get- Command () ECRLoginAWS Tools for Windows PowerShell

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Uso de la autenticación de la API HTTP

Amazon ECR es compatible con la API HTTP de Docker Registry. No obstante, dado que Amazon ECR es un registro privado, debe proporcionar un token de autorización con cada solicitud HTTP. Puede añadir un encabezado de autorización HTTP mediante la `-H` opción for curl y pasar el token de autorización proporcionado por el get-authorization-token AWS CLI comando.

Autenticación con la API HTTP de Amazon ECR

1. Recupera un token de autorización con AWS CLI y configúralo en una variable de entorno.

```
TOKEN=$(aws ecr get-authorization-token --output text --query
 'authorizationData[].authorizationToken')
```

2. Para autenticar en la API, pase la variable `$TOKEN` a la opción `-H` de curl. Por ejemplo, el siguiente comando muestra las etiquetas de imagen de un repositorio de Amazon ECR. Para obtener más información, consulte la documentación de referencia de la API HTTP de Docker Registry.

```
curl -i -H "Authorization: Basic $TOKEN"
 https://aws_account_id.dkr.ecr.region.amazonaws.com/v2/amazonlinux/tags/list
```

La salida es la siguiente:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
Connection: keep-alive

{"name":"amazonlinux","tags":["2017.09","latest"]}
```

Configuración del registro privado en Amazon ECR

Amazon ECR utiliza configuraciones de registro privado para configurar características de registros. La configuración del registro privado se establece de manera independiente para cada región. Puede utilizar la configuración de registro privado para configurar las siguientes características.

- Permisos de registro: una política de permisos de registro proporciona control sobre la replicación y la extracción de los permisos de caché. Para obtener más información, consulte Permisos de registro privado en Amazon ECR.
- Reglas de extracción de memoria caché: se utiliza una regla de extracción de memoria caché para almacenar en caché imágenes de un registro anterior en su registro privado de Amazon ECR. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.
- Configuración de replicación: la configuración de replicación se utiliza para controlar si los repositorios se copian entre AWS regiones o cuentas. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR
- Plantillas de creación de repositorios: se utiliza una plantilla de creación de repositorios para definir la configuración estándar que se aplicará cuando Amazon ECR cree nuevos repositorios en su nombre. Por ejemplo, los repositorios creados mediante una acción de caché de extracción. Para obtener más información, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.

- Configuración de escaneo: de forma predeterminada, su registro está habilitado para el escaneo básico. Puede habilitar el escaneo mejorado, el cual proporciona un modo de escaneo automático y continuo que analiza las vulnerabilidades del paquete de idiomas de programación y del sistema operativo. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.

Permisos de registro privado en Amazon ECR

Amazon ECR utiliza una política de registro para conceder permisos a un principal de AWS a nivel de registro privado.

El alcance se establece al elegir la versión de la política de registro. Hay dos versiones con un ámbito de política de registro diferente: la versión 1 (V1) y la versión 2 (V2). La versión 2 es el ámbito ampliado de la política de registro que incluye todos los permisos de ECR. Para ver la lista completa de acciones de API, consulte la Guía de API de Amazon ECR. La versión V2 es el ámbito de aplicación de la política de registro predeterminado. Para obtener más información sobre cómo ver o configurar el alcance de su política de registro, consulteCambiar al ámbito ampliado de la política de registro. Para obtener información sobre la configuración general de su registro privado de Amazon ECR, consulteConfiguración del registro privado en Amazon ECR.

Las versiones se detallan a continuación.

- V1: para la versión 1, Amazon ECR solo aplica los siguientes permisos en el nivel de registro privado.
  - `ecr:ReplicateImage`: otorga permiso a otra cuenta, denominada registro de origen, para replicar sus imágenes en el registro. Solo se utiliza para la replicación entre cuentas.
  - `ecr:BatchImportUpstreamImage`: otorga permiso para recuperar la imagen externa e importarla a su registro privado.
  - `ecr:CreateRepository`: otorga permiso para crear un repositorio en un registro privado. Este permiso es necesario si el repositorio donde se conservan las imágenes replicadas o almacenadas en caché no existe todavía en el registro privado.
- V2: para la versión 2, Amazon ECR permite todas las acciones de ECR en la política y aplica la política de registro en todas las solicitudes de ECR.

Puede usar la consola o la CLI para ver o cambiar el alcance de la política de registro.

> **Note**
>
> Si bien es posible agregar la `ecr:*` acción a una política de registro privada, se considera una buena práctica agregar solo las acciones específicas necesarias en función de la función que esté utilizando, en lugar de usar un comodín.

Temas

- Ejemplos de políticas de registro privado para Amazon ECR
- Cambiar al ámbito ampliado de la política de registro
- Concesión de permisos de registro para la replicación entre cuentas en Amazon ECR
- Concesión de permisos de registro para caché de extracción en Amazon ECR

Ejemplos de políticas de registro privado para Amazon ECR

En los siguientes ejemplos se muestran instrucciones de política de permisos de registro que puede utilizar para controlar los permisos que los usuarios tienen en el registro de Amazon ECR.

> **Note**
>
> En cada ejemplo, si la `ecr:CreateRepository` acción se elimina de la política de registro, se puede seguir produciendo la replicación. No obstante, para que la replicación se realice correctamente, debe crear repositorios con el mismo nombre en su cuenta.

Ejemplo: Permitir al usuario raíz de una cuenta de origen replicar todos los repositorios

La siguiente política de permisos de registro permite al usuario raíz de una cuenta de origen replicar todos los repositorios.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_id:root"
```

```
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/*"
            ]
        }
    ]
}
```

Ejemplo: permitir que los usuarios raíz pertenezcan a varias cuentas

La siguiente política de permisos de registro tiene dos instrucciones. Cada instrucción permite al usuario raíz de una cuenta de origen replicar todos los repositorios.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount1",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_1_id:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/*"
            ]
        },
        {
            "Sid":"ReplicationAccessCrossAccount2",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_2_id:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
```

```
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/*"
            ]
        }
    ]
}
```

Ejemplo: Permitir al usuario raíz de una cuenta de origen replicar todos los repositorios con el prefijo **`prod-`**.

La siguiente política de permisos de registro permite al usuario raíz de una cuenta de origen replicar todos los repositorios que comiencen con `prod-`.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_id:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/prod-*"
            ]
        }
    ]
}
```

Cambiar al ámbito ampliado de la política de registro

> **Important**
>
> En el caso de los nuevos usuarios, los registros se configuran automáticamente para utilizar la política de V2 registro al crearlos. No hay ninguna acción que deba realizar. Amazon ECR no recomienda volver a la política de registro anterior. V1

Puede usar la consola o la CLI para ver o cambiar el alcance de la política de registro.

AWS Management Console

Siga los pasos siguientes para ver la configuración de su cuenta. Para ver o actualizar el alcance de la política de registro, consulte el procedimiento CLI de esta página.

Active la política de registro mejorada para su registro privado

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/ private-registry/ repositories
2. En la barra de navegación, seleccione la región.
3. En el panel de navegación, selecciona Registro privado, función y configuración y, a continuación, selecciona Permisos.
4. En la página de permisos, para ver la política de registro, consulta el JSON de tu política. Si tienes la política V1, aparecerá un banner con instrucciones para actualizar a la V2. Seleccione Habilitar.

   Aparece un banner que indica que el alcance de la política de registro se ha actualizado a la versión 2.
5. Opcionalmente, también puede configurar los permisos con la CLI. Para obtener más información, consulte Configuración del registro privado en Amazon ECR.

> **Note**
>
> Para ver o actualizar el alcance de la política de registro, consulte el procedimiento CLI de esta página.

AWS CLI

Amazon ECR genera la política de registro de la versión 2. Siga los pasos siguientes para ver o actualizar el alcance de la política de registro. No puede ver ni cambiar el alcance de la política de registro en la consola

- Para recuperar la política de registro que está utilizando actualmente.

```
aws ecr get-account-setting --name REGISTRY_POLICY_SCOPE
```

El campo del nombre del parámetro es obligatorio. Si no proporciona el nombre, recibirá el siguiente mensaje de error:

```
 aws: error: the following arguments are required: --name
```

Vea el resultado del comando de política de registro. En el siguiente resultado de ejemplo, la versión de la política de registro es la V1.

```
{
    "name": "REGISTRY_POLICY_SCOPE",
    "value": "V1"
}
```

Puede cambiar la versión de la política de registro de V1 aV2. La versión 1 no es el ámbito de aplicación recomendado de la política de registro.

```
aws ecr put-account-setting --name REGISTRY_POLICY_SCOPE --value value
```

Por ejemplo, utilice el siguiente comando para actualizar a la versión 2.

```
aws ecr put-account-setting --name REGISTRY_POLICY_SCOPE --value V2
```

Vea el resultado del comando de política de registro. En el siguiente resultado de ejemplo, la versión de la política de registro se actualizó a la versión 2.

```
{
    "name": "REGISTRY_POLICY_SCOPE",
    "value": "V2"
```

```
}
```

Concesión de permisos de registro para la replicación entre cuentas en Amazon ECR

El tipo de política multicuenta se utiliza para conceder permisos a una entidad AWS principal, lo que permite replicar los repositorios de un registro de origen al suyo. De forma predeterminada, tiene permiso para configurar la replicación entre regiones en su propio registro. Tan solo necesita configurar la política de registro si concede permiso a otra cuenta para replicar contenido en el registro.

Una política de registro debe conceder permiso para la acción de la API `ecr:ReplicateImage`. Esta es una API de Amazon ECR interna que puede replicar imágenes entre Regiones o cuentas. También puede concederlo para el permiso `ecr:CreateRepository`, que permite a Amazon ECR crear repositorios en el registro si aún no existen. Si no se proporciona el permiso de `ecr:CreateRepository`, en el registro se debe crear manualmente un repositorio con el mismo nombre que el de origen. Si no se hace nada de esto, la replicación fallará. Todas las acciones fallidas `CreateRepository` o de la `ReplicateImage` API aparecen en CloudTrail.

Para configurar una política de permisos para la replicación (AWS Management Console)

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que se va a configurar la política de registro.
3. En el panel de navegación, elija Registro privado, elija Características y configuración y, a continuación, elija Permisos.
4. En la página Registry permissions (Permisos de registro), elija Generate statement (Generar declaración).
5. Realice los siguientes pasos para definir la instrucción de política mediante el generador de políticas.

    a. Para el tipo de política, elija Replicación: cuenta cruzada.

    b. En el campo ID del estado de cuenta, introduzca un identificador de estado de cuenta único. Este campo se utiliza como `Sid` en la política de registro.

    c. En el caso de las cuentas, introduzca IDs la cuenta de cada cuenta a la que desee conceder permisos. Al especificar varias cuentas IDs, sepárelas con una coma.
6. Seleccione Save.

Para configurar una política de permisos para la replicación (AWS CLI)

1. Cree un archivo denominado `registry_policy.json` y rellénelo con una política de registro.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_id:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/*"
            ]
        }
    ]
}
```

2. Cree la política de registro mediante el archivo de política.

```
aws ecr put-registry-policy \
      --policy-text file://registry_policy.json \
      --region us-west-2
```

3. Recupere la política de registro que se va a confirmar.

```
aws ecr get-registry-policy \
      --region us-west-2
```

Concesión de permisos de registro para caché de extracción en Amazon ECR

Los permisos de registro privado de Amazon ECR se pueden utilizar para abarcar los permisos de entidades de IAM individuales a fin de utilizar caché de extracción. Si una entidad de IAM tiene más

permisos otorgados por una política de IAM de los que la política de permisos de registro concede, prevalece la política de IAM.

Para crear una política de permisos de registro privado (AWS Management Console):

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que configurar su declaración de permisos de registro privado.
3. En el panel de navegación, elija Registro privado, elija Características y configuración y, a continuación, elija Permisos.
4. En la página Registry permissions (Permisos de registro), elija Generate statement (Generar declaración) .
5. Para cada declaración de política de permisos de caché de extracción que desee crear, haga lo siguiente.

    a. Para Policy type (Tipo de política), elija Pull through cache policy (Política de caché de extracción).
    b. Para Statement id (ID de declaración), proporcione un nombre para la política de declaración de caché de extracción.
    c. Para AM entities (Entidades de IAM), especifique los usuarios, grupos o roles que se deben incluir en la política.
    d. Para el espacio de nombres de caché, selecciona la regla de extracción de caché a la que deseas asociar la política.
    e. Para Repository names (Nombres de repositorio), especifique el nombre base del repositorio para el que se va a aplicar la regla. Por ejemplo, si desea especificar el repositorio de Amazon Linux en Amazon ECR Public, el nombre del repositorio sería `amazonlinux`.

Repositorios privados de Amazon ECR

Un repositorio privado de Amazon ECR contiene las imágenes de Docker, las imágenes de Iniciativa de contenedores abiertos (OCI) y artefactos compatibles con OCI. Puede crear, monitorear y eliminar repositorios de imágenes, así como definir permisos que controlen quién puede acceder a ellos mediante las operaciones de API de Amazon ECR o la sección de Repositorios de la consola de Amazon ECR. Amazon ECR se integra también con la CLI de Docker, de modo que pueda insertar y extraer imágenes de los entornos de desarrollo a los repositorios.

Temas

- Conceptos sobre los repositorios privados
- Creación de un repositorio privado de Amazon ECR para almacenar imágenes
- Visualización del contenido y los detalles de un repositorio privado en Amazon ECR
- Eliminación de un repositorio privado en Amazon ECR
- Políticas de repositorios privados en Amazon ECR
- Etiquetado de un repositorio privado en Amazon ECR

Conceptos sobre los repositorios privados

- De forma predeterminada, su cuenta tiene acceso de lectura y escritura en los repositorios e imágenes que cree en el registro predeterminado (*aws_account_id*.dkr.ecr.*region*.amazonaws.com). Sin embargo, los usuarios necesitan permisos para realizar llamadas al Amazon ECR APIs y para enviar o extraer imágenes de sus repositorios. Amazon ECR proporciona varias políticas administradas por para controlar el acceso de los usuarios en distintos niveles. Para obtener más información, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.
- Los repositorios se pueden controlar mediante políticas tanto de acceso de usuarios de como de repositorio individuales. Para obtener más información, consulte Políticas de repositorios privados en Amazon ECR.
- Los nombres de repositorio pueden admitir espacios de nombres, que puede usar para agrupar repositorios similares. Por ejemplo, si varios equipos usan el mismo registro, el equipo A puede usar el espacio de nombres `team-a` y el equipo B puede usar el espacio de nombres `team-b`. Al hacer esto, cada equipo tiene su propia imagen, llamada `web-app`, con cada imagen precedida por el espacio de nombres del equipo. Esta configuración permite que estas imágenes en cada

equipo se utilicen simultáneamente sin interferencias. La imagen del equipo A es `team-a/web-app` y la del equipo B es `team-b/web-app`.

- Las imágenes se pueden replicar en otros repositorios entre regiones, en su propio registro, y entre cuentas. Para ello, especifique una configuración de replicación en las opciones del registro. Para obtener más información, consulte Configuración del registro privado en Amazon ECR.

Creación de un repositorio privado de Amazon ECR para almacenar imágenes

> **Important**
>
> El cifrado de doble capa del lado del servidor con AWS KMS (DSSE-KMS) solo está disponible en las regiones. AWS GovCloud (US)

Cree un repositorio privado de Amazon ECR y, a continuación, utilícelo para almacenar sus imágenes de contenedor. Utilice los siguientes pasos para crear un repositorio privado mediante la AWS Management Console. Para conocer los pasos para crear un repositorio mediante el AWS CLI, consulte. Paso 2: crear un repositorio

Creación de un repositorio (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, elija la región en la que va a crear el repositorio.
3. Elija Repositorios privados y, a continuación, elija Crear repositorio.
4. En Repository name (Nombre del repositorio), ingrese un nombre exclusivo para el repositorio. El nombre del repositorio se puede especificar por sí mismo (por ejemplo, `nginx-web-app`). También puede ir precedido de un espacio de nombres para agrupar el repositorio en una categoría (por ejemplo, `project-a/nginx-web-app`).

   > **Note**
   >
   > El nombre del repositorio puede contener un máximo de 256 caracteres. Los nombres deben comenzar por una letra y solo pueden contener letras minúsculas, números,

guiones, guiones bajos, puntos y barras inclinadas. No se admite el uso de guiones dobles, guiones bajos o barras oblicuas.

5. Para la inmutabilidad de las etiquetas de imagen, elige una de las siguientes configuraciones de mutabilidad de etiquetas para el repositorio.

   - Mutable: elija esta opción si desea sobrescribir las etiquetas de imagen. Se recomienda para los repositorios que utilizan acciones de extracción de caché para garantizar que Amazon ECR pueda actualizar las imágenes almacenadas en caché. Además, para deshabilitar las actualizaciones de etiquetas para algunas etiquetas mutables, introduzca los nombres de las etiquetas o utilice caracteres comodín (*) para hacer coincidir varias etiquetas similares en el cuadro de texto de exclusión de etiquetas mutables.
   - Inmutable: elige esta opción si quieres evitar que las etiquetas de las imágenes se sobrescriban y se aplica a todas las etiquetas y exclusiones del repositorio al insertar una imagen con una etiqueta existente. Amazon ECR devuelve un `ImageTagAlreadyExistsException` si intentas insertar una imagen con una etiqueta existente. Además, para habilitar las actualizaciones de etiquetas para algunas etiquetas inmutables, introduzca los nombres de las etiquetas o utilice caracteres comodín (*) para hacer coincidir varias etiquetas similares en el cuadro de texto de exclusión de etiquetas inmutables.

   > **Note**
   >
   > No se admite la configuración de mutabilidad de etiquetas individuales.

6. Para configurar el cifrado, elija entre AES-256 o. AWS KMS Para obtener más información, consulte Cifrado en reposo.

   a. Si AWS KMS se elige, elija entre el cifrado de una capa y el cifrado de doble capa. El uso AWS KMS del cifrado de doble capa conlleva cargos adicionales. Para obtener más información, consulte los Precios del servicio de Amazon ECR.

   b. De forma predeterminada, se elige la clave AWS administrada con `aws/ecr` el alias. Esta clave se crea en tu cuenta la primera vez que creas un repositorio con el AWS KMS cifrado activado. Seleccione Clave gestionada por el cliente (avanzada) para elegir su propia clave AWS KMS . La AWS KMS clave debe estar en la misma región que el clúster. Seleccione Crear una AWS KMS clave para ir a la AWS KMS consola y crear su propia clave.

7. En cuanto a la configuración de digitalización de imágenes, si bien puede especificar la configuración de digitalización a nivel de repositorio para la digitalización básica, se recomienda especificar la configuración de digitalización a nivel de registro privado. Configurar el escaneo en el registro privado le permite elegir entre escaneo mejorado o escaneo básico, así como definir filtros para especificar qué repositorios se deben escanear.
8. Seleccione Crear.

Pasos a seguir a continuación

Seleccione el repositorio y elija Ver comandos de inserción para ver los pasos para insertar una imagen en su nuevo repositorio. Para obtener más información sobre cómo insertar una imagen en el repositorio, consulte Inserción de una imagen en un repositorio privado de Amazon ECR.

Visualización del contenido y los detalles de un repositorio privado en Amazon ECR

Una vez que haya creado un repositorio privado, puede ver sus detalles en la AWS Management Console:

- Qué imágenes están almacenadas en un repositorio
- Detalles de cada imagen almacenada en el repositorio, como el tamaño y el resumen de SHA de cada una
- Frecuencia de análisis especificada para el contenido del repositorio
- Si el repositorio tiene asociada una regla de caché de extracción activa
- Configuración de cifrado del repositorio

> **Note**
>
> A partir de la versión de Docker 1.9, el cliente Docker comprime las capas de las imágenes antes de insertarlas en un registro V2 Docker. La salida del comando docker images muestra el tamaño de la imagen sin comprimir. Por lo tanto, tenga en cuenta que Docker podría devolver una imagen más grande que la mostrada en la AWS Management Console.

Visualización de la información del repositorio (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, seleccione la región que contiene el repositorio que desea ver.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios), elija la pestaña Private (Privados) y, a continuación, el repositorio que desea ver.
5. En la página de detalles del repositorio, la consola muestra de forma predeterminada la vista Images (Imágenes). Utilice el menú de navegación para ver otro tipo de información sobre el repositorio.
   - Elija Summary (Resumen) para ver los detalles del repositorio y los datos del recuento de extracción del repositorio.
   - Elija Images (Imágenes) para ver información sobre las etiquetas de las imágenes del repositorio. Para ver más información sobre la imagen, seleccione la etiqueta de la imagen. Para obtener más información, consulte Visualización de los detalles de una imagen en Amazon ECR.

     Si hay imágenes sin etiquetar que desea eliminar, puede seleccionar la casilla situada la izquierda de los repositorios que desea eliminar y elegir Delete (Eliminar). Para obtener más información, consulte Eliminación de una imagen en Amazon ECR.
   - Elija Permissions (Permisos) para ver las políticas de repositorio que se aplican al repositorio. Para obtener más información, consulte Políticas de repositorios privados en Amazon ECR.
   - Elija Lifecycle Policy (Política de ciclo de vida) para ver las reglas de política de ciclo de vida que se aplican al repositorio. El historial de eventos del ciclo de vida también se visualiza aquí. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.
   - Elija Tags (Etiquetas) para ver las etiquetas de metadatos que se aplican al repositorio.

Eliminación de un repositorio privado en Amazon ECR

Si ya ha terminado de usar un repositorio, puede eliminarlo. Al eliminar un repositorio del AWS Management Console, también se eliminan todas las imágenes contenidas en el repositorio; esto no se puede deshacer.

> **Important**
>
> Se eliminan también las imágenes de los repositorios eliminados. No podrá deshacer esta operación.

Para eliminar un repositorio (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, seleccione la región que contiene el repositorio que desea eliminar.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios), elija la pestaña Private (Privados) y, a continuación, seleccione el repositorio que desea eliminar y elija Delete (Eliminar).
5. En la ***repository_name*** ventana Eliminar, compruebe que se deben eliminar los repositorios seleccionados y elija Eliminar.

Políticas de repositorios privados en Amazon ECR

Amazon ECR utiliza permisos basados en recursos para controlar el acceso a los repositorios. Los permisos basados en recursos le permiten especificar qué usuarios o roles tienen acceso a un repositorio y qué acciones pueden realizar en él. De forma predeterminada, solo la AWS cuenta que creó el repositorio tiene acceso al repositorio. Puede aplicar una política de repositorio que otorgue acceso adicional a su repositorio.

Temas

- Políticas de repositorios frente a políticas de IAM
- Ejemplos de políticas de repositorio privado en Amazon ECR
- Configuración de una instrucción de política de repositorio privado en Amazon ECR

Políticas de repositorios frente a políticas de IAM

Las políticas de repositorios de Amazon ECR son un subconjunto de políticas de IAM destinados a controlar el acceso a repositorios de Amazon ECR individuales, y se aplican específicamente a este fin. Las políticas de IAM se suelen utilizar para aplicar permisos a todo el servicio de Amazon ECR, pero también se pueden utilizar para controlar el acceso a recursos específicos.

Tanto las políticas de repositorios de Amazon ECR como las políticas de IAM se utilizan a la hora de determinar qué acciones puede realizar un rol o usuario específico en un repositorio. Si se le permite realizar una acción a un rol o usuario mediante una política de repositorio, pero el permiso se deniega mediante una política de IAM (o viceversa), la acción se denegará. Para que un usuario o rol pueda realizar una acción, solo necesita que se le proporcione permiso mediante una política de repositorio o una política de IAM, no de ambas.

> **Important**
>
> Amazon ECR requiere que los usuarios tengan permiso para realizar llamadas a la API `ecr:GetAuthorizationToken` a través de una política de IAM para que puedan autenticarse en un registro, así como insertar o extraer imágenes de cualquier repositorio de Amazon ECR. Amazon ECR proporciona varias políticas de IAM gestionadas para controlar el acceso de los usuarios en distintos niveles. Para obtener más información, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.

Puede utilizar cualquiera de estos tipos de política para controlar el acceso a los repositorios, como se muestra en los siguientes ejemplos.

En este ejemplo se muestra una política de repositorio de Amazon ECR que permite a un usuario específico describir el repositorio y las imágenes que contiene este.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECRRepositoryPolicy",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::account-id:user/username"},
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ]
        }
    ]
}
```

En este ejemplo se muestra una política de IAM que logra el mismo objetivo que la anterior al limitar la política a un repositorio (especificado por el ARN completo del repositorio) mediante el parámetro

de recurso. Para obtener más información sobre el formato de Nombres de recursos de Amazon (ARN), consulte Recursos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeRepoImage",
            "Effect": "Allow",
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": ["arn:aws:ecr:region:account-id:repository/repository-name"]
        }
    ]
}
```

Ejemplos de políticas de repositorio privado en Amazon ECR

> **Important**
>
> Los ejemplos de políticas de repositorios de esta página están pensados para aplicarse a los repositorios privados de Amazon ECR. No funcionarán correctamente si se utilizan directamente con una entidad principal de IAM, a menos que se modifiquen para especificar el repositorio de Amazon ECR como recurso. Para obtener más información acerca de cómo establecer políticas de repositorios, consulte Configuración de una instrucción de política de repositorio privado en Amazon ECR.

Las políticas de repositorios de Amazon ECR son un subconjunto de políticas de IAM destinados a controlar el acceso a repositorios de Amazon ECR individuales, y se aplican específicamente a este fin. Las políticas de IAM se suelen utilizar para aplicar permisos a todo el servicio de Amazon ECR, pero también se pueden utilizar para controlar el acceso a recursos específicos. Para obtener más información, consulte Políticas de repositorios frente a políticas de IAM.

En los siguientes ejemplos de política de repositorio se muestran instrucciones de permiso que puede utilizar para controlar el acceso a los repositorios privados de Amazon ECR.

> **Important**
>
> Amazon ECR requiere que los usuarios tengan permiso para realizar llamadas a la API `ecr:GetAuthorizationToken` a través de una política de IAM para que puedan autenticarse en un registro, así como insertar o extraer imágenes de cualquier repositorio de Amazon ECR. Amazon ECR proporciona varias políticas de IAM gestionadas para controlar el acceso de los usuarios en distintos niveles. Para obtener más información, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.

Ejemplo: Permitir uno o más usuarios de

La siguiente política de repositorio permite que uno o más usuarios de inserten imágenes en un repositorio y las extraigan de este.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPushPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:user/push-pull-user-1",
                    "arn:aws:iam::account-id:user/push-pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:GetDownloadUrlForLayer",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ]
        }
    ]
}
```

Ejemplo: Permitir otra cuenta

La siguiente política de repositorio permite a una cuenta específica insertar imágenes.

> **Important**
>
> La cuenta a la que está otorgando permisos debe tener habilitada la región en la que está creando la política de repositorio; de lo contrario, se producirá un error.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPush",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:root"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ]
        }
    ]
}
```

La siguiente política de repositorios permite a algunos usuarios extraer imágenes (*pull-user-1*y*pull-user-2*) y, al mismo tiempo, proporcionar acceso total a otros (*admin-user*).

> **Note**
>
> Para políticas de repositorio más complicadas que actualmente no son compatibles con el AWS Management Console, puedes aplicar la política con el set-repository-policy AWS CLI comando.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:user/pull-user-1",
                    "arn:aws:iam::account-id:user/pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        },
        {
            "Sid": "AllowAll",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:user/admin-user"
            },
            "Action": [
                "ecr:*"
            ]
        }
    ]
}
```

Ejemplo: Denegar todo

La siguiente política de repositorio deniega a todos los usuarios de todas las cuentas la capacidad de extraer imágenes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPull",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
```

```
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Ejemplo: Restringir el acceso a direcciones IP específicas

El siguiente ejemplo deniega permisos a cualquier usuario para realizar cualquier operación de Amazon ECR cuando se aplica a un repositorio desde un rango específico de direcciones.

La condición de esta declaración identifica el `54.240.143.*` rango de direcciones IP del Protocolo de Internet de la versión 4 (IPv4) permitidas.

El `Condition` bloque usa las `NotIpAddress` condiciones y la clave de `aws:SourceIp` condición, que es una clave AWS de condición amplia. Para obtener más información sobre estas claves de condición, consulte Claves de contexto de condición globales de AWS. Los `aws:sourceIp` IPv4 valores utilizan la notación CIDR estándar. Para obtener más información, consulte Operadores de condición de dirección IP en la guía del usuario de IAM.

```
{
    "Version": "2012-10-17",
    "Id": "ECRPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "ecr:*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "54.240.143.0/24"
                }
            }
        }
    ]
}
```

Ejemplo: permitir un servicio AWS

La siguiente política de repositorios permite el AWS CodeBuild acceso a las acciones de la API Amazon ECR necesarias para la integración con ese servicio. Al utilizar el siguiente ejemplo, debe utilizar `aws:SourceArn` y las claves de condición `aws:SourceAccount` para el ámbito de qué recursos pueden asumir estos permisos. Para obtener más información, consulte el ejemplo de Amazon ECR CodeBuild en la Guía del AWS CodeBuild usuario.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:region:123456789012:project/project-
name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

Configuración de una instrucción de política de repositorio privado en Amazon ECR

Puede añadir una declaración de política de acceso a un repositorio AWS Management Console siguiendo los pasos que se indican a continuación. Puede añadir varias instrucciones de política por repositorio. Para ver ejemplos de políticas, consulte Ejemplos de políticas de repositorio privado en Amazon ECR.

> **Important**
>
> Amazon ECR requiere que los usuarios tengan permiso para realizar llamadas a la API `ecr:GetAuthorizationToken` a través de una política de IAM para que puedan autenticarse en un registro, así como insertar o extraer imágenes de cualquier repositorio de Amazon ECR. Amazon ECR proporciona varias políticas de IAM gestionadas para controlar el acceso de los usuarios en distintos niveles. Para obtener más información, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.

Para definir una instrucción de política de repositorio

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, seleccione la región que contiene el repositorio para el que desea definir una instrucción de política.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios), elija el repositorio para el que desea definir una instrucción de política con objeto de ver su contenido.
5. En la vista de lista de imágenes del repositorio, en el panel de navegación, elija Permissions (Permisos), Edit (Editar).

   > **Note**
   >
   > Si no ve la opción Permissions (Permisos) en el panel de navegación, asegúrese de que está en la vista de lista de imágenes del repositorio.

6. En la página Edit permissions (Editar permisos), seleccione Add statement (Añadir declaración).
7. En Statement name (Nombre de la declaración), introduzca un nombre para la declaración.
8. En Effect (Efecto), elija si desea que la instrucción de política de como resultado un permiso o una denegación explícita.
9. En Principal (Entidad de seguridad), elija el alcance con el que se va a aplicar la instrucción de la política. Para obtener más información, consulte Elementos de la política de JSON de AWS : Entidad de servicio en la Guía del usuario de IAM.

   - Puede aplicar la declaración a todos los AWS usuarios autenticados seleccionando la casilla Todos (*).

- En (Entidad de servicio), especifique el nombre de la entidad de servicio (por ejemplo, `ecs.amazonaws.com`) para aplicar la instrucción a un servicio específico.
- IDsEn AWS Cuenta, especifique un número de AWS cuenta (por ejemplo`111122223333`) para aplicar la declaración a todos los usuarios de una AWS cuenta específica. Se pueden especificar varias cuentas utilizando una lista delimitada por comas.

> **Important**
>
> La cuenta a la que está otorgando permisos debe tener habilitada la región en la que está creando la política de repositorio; de lo contrario, se producirá un error.

- En el caso de las entidades de IAM, seleccione los roles o usuarios de su AWS cuenta a los que desee aplicar la declaración.

> **Note**
>
> Para políticas de repositorio más complicadas que actualmente no son compatibles con el AWS Management Console, puede aplicar la política con el set-repository-policy AWS CLI comando.

10. En Actions (Acciones), elija el alcance de las operaciones de la API de Amazon ECR a las que debe aplicarse la instrucción de la política en la lista de operaciones de API individuales.
11. Cuando haya terminado, elija Save para definir la política.
12. Repita el paso anterior para cada política de repositorio que desee añadir.

Etiquetado de un repositorio privado en Amazon ECR

Para ayudarle a gestionar sus repositorios de Amazon ECR, puede asignar sus propios metadatos a los repositorios de Amazon ECR nuevos o existentes mediante etiquetas de recursos. AWS Por ejemplo, puede definir un conjunto de etiquetas para los repositorios de Amazon ECR de su cuenta que le ayuden a rastrear al propietario de cada repositorio.

Conceptos básicos de etiquetas

Las etiquetas no tienen ningún significado semántico para Amazon ECR, por lo que se interpretan estrictamente como cadenas de caracteres. Además, las etiquetas no se asignan a los recursos automáticamente. Puede editar las claves y los valores de las etiquetas y también puede eliminar

etiquetas de un recurso en cualquier momento. Puede establecer el valor de una etiqueta como una cadena vacía, pero no puede asignarle un valor nulo. Si añade una etiqueta con la misma clave que una etiqueta existente en ese recurso, el nuevo valor sobrescribirá al antiguo. Si elimina un recurso, también se eliminará cualquier etiqueta asignada a dicho recurso.

Puede trabajar con etiquetas mediante la consola Amazon ECR AWS CLI, la y la API Amazon ECR.

Con AWS Identity and Access Management (IAM), puede controlar qué usuarios de su AWS cuenta tienen permiso para crear, editar o eliminar etiquetas. Para obtener información acerca de las etiquetas en la política de IAM, consulte the section called “Uso del control de acceso basado en etiquetas”.

Etiquetado de los recursos para facturación

Las etiquetas que agregue a sus repositorios de Amazon ECR son útiles a la hora de revisar la asignación de costes una vez que las habilite en el informe de uso y costes. Para obtener más información, consulte Informes de uso de Amazon ECR.

Para ver el costo de los recursos combinados, puede organizar la información de facturación basada en los recursos que tienen los mismos valores de clave de etiqueta. Por ejemplo, puede etiquetar varios recursos con un nombre de aplicación específico y luego organizar su información de facturación para ver los costos totales de la aplicación en distintos servicios. Para obtener más información sobre la configuración de un informe de asignación de costes con etiquetas, consulte Informe de asignación de costes mensual en la Guía del usuario de AWS Billing .

> **Note**
>
> Si acaba de habilitar la realización de informes, los datos correspondientes al mes actual estarán disponibles para su visualización transcurridas 24 horas.

Incorporación de etiquetas a un repositorio privado en Amazon ECR

Puede agregar etiquetas a un repositorio privado.

Para obtener información sobre los nombres y las prácticas recomendadas para las etiquetas, consulte los límites y requisitos de nomenclatura de etiquetas y las prácticas recomendadas en la Guía del usuario de Tagging AWS Resources.

Incorporación de etiquetas a un repositorio (AWS Management Console)

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, seleccione la región que desea utilizar.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios), active la casilla de verificación situada junto al repositorio que desee etiquetar.
5. En el menú Action (Acción), selecciona Repository tags (Etiquetas de repositorio).
6. En la pestaña Repository tags (Etiquetas de repositorio), seleccione Add tags (Añadir etiquetas), Add tag (Añadir etiqueta).
7. En la página Edit repository tags (Editar etiquetas de repositorio), especifique la clave y el valor de cada etiqueta y, a continuación, elija save (Guardar).

Añadir etiquetas a un repositorio (AWS CLI o API)

Puedes añadir o sobrescribir una o más etiquetas mediante la AWS CLI o una API.

- AWS CLI - recurso de etiquetas
- Acción de la API - TagResource

En los siguientes ejemplos se muestra cómo agregar etiquetas mediante la función AWS CLI.

Ejemplo 1: etiquetar un repositorio

El siguiente comando etiqueta un repositorio.

```
aws ecr tag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tags Key=stack,Value=dev
```

Ejemplo 2: etiquetar un repositorio con varias etiquetas

El siguiente comando agrega tres etiquetas a un repositorio.

```
aws ecr tag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tags Key=key1,Value=value1 Key=key2,Value=value2 Key=key3,Value=value3
```

Ejemplo 3: Enumerar etiquetas de un repositorio

El siguiente comando enumera las etiquetas asociadas a un repositorio.

```
aws ecr list-tags-for-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name
```

Ejemplo 4: crear un repositorio y aplicar una etiqueta

El siguiente comando crea un repositorio denominado `test-repo` y añade una etiqueta con clave `team` y valor `devs`.

```
aws ecr create-repository \
     --repository-name test-repo \
     --tags Key=team,Value=devs
```

Eliminación de etiquetas de un repositorio privado en Amazon ECR

Puede eliminar etiquetas de un repositorio privado.

Para eliminar una etiqueta de un repositorio privado (AWS Management Console)

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, seleccione la región que desea utilizar.
3. En la página Repositories (Repositorios), active la casilla de verificación situada junto al repositorio del que desee eliminar una etiqueta.
4. En el menú Action (Acción), selecciona Etiquetas de repositorio.
5. En la página Repository tags (Etiquetas de repositorio), seleccione Edit (Editar).
6. En la página Edit repository tags (Editar etiquetas de repositorio), seleccione Remove (Eliminar) para cada etiqueta que desee eliminar y, a continuación, elija Save (Guardar).

Para eliminar una etiqueta de un repositorio privado (AWS CLI)

Puede eliminar una o más etiquetas mediante la AWS CLI o una API.

- AWS CLI - Recurso sin etiquetar
- Acción de la API - UntagResource

En el siguiente ejemplo se muestra cómo eliminar una etiqueta de un repositorio con la ayuda de AWS CLI.

```
aws ecr untag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tag-keys tag_key
```

Imágenes privadas en Amazon ECR

Amazon ECR almacena imágenes de Docker, imágenes de Iniciativa de contenedores abiertos (OCI) y artefactos compatibles con OCI en repositorios privados. Puede utilizar la CLI de Docker, o su cliente favorito, para insertar imágenes de sus repositorios y extraerlas desde estos.

Gracias a la compatibilidad de Amazon ECR con OCI v1.1, puede almacenar y gestionar los artefactos de referencia definidos por la API de referencia de OCI. Los artefactos incluyen firmas, listas de materiales (SBoMS) de software, gráficos de Helm, resultados de escaneos y atestaciones. El conjunto de artefactos de una imagen de contenedor se transfiere con ese mismo contenedor y se almacena como una imagen independiente que se considera una imagen consumida por el repositorio.

Las páginas Firma de una imagen almacenada en un repositorio privado de Amazon ECR y Eliminación de firmas y otros artefactos de un repositorio privado de Amazon ECR proporcionan ejemplos de cómo utilizar los artefactos relacionados con las firmas. Para obtener más información acerca de la firma de imágenes de contenedores, consulte Signing container images en la Guía para desarrolladores de AWS Signer .

Temas

- Inserción de una imagen en un repositorio privado de Amazon ECR
- Firma de una imagen almacenada en un repositorio privado de Amazon ECR
- Eliminación de firmas y otros artefactos de un repositorio privado de Amazon ECR
- Visualización de los detalles de una imagen en Amazon ECR
- Extracción de una imagen a su entorno local desde un repositorio privado de Amazon ECR
- Extracción de la imagen de contenedor Linux de Amazon
- Eliminación de una imagen en Amazon ECR
- Reetiquetación de una imagen en Amazon ECR
- Cómo impedir que se sobrescriban las etiquetas de imagen en Amazon ECR
- Formatos de manifiesto de imagen de contenedor compatibles con Amazon ECR
- Uso de imágenes de Amazon ECR con Amazon ECS
- Uso de imágenes de Amazon ECR con Amazon EKS

Inserción de una imagen en un repositorio privado de Amazon ECR

Puede insertar las imágenes Docker, las listas de manifiesto, las imágenes de Open Container Initiative (OCI) y los artefactos compatibles con OCI a sus repositorios privados.

Amazon ECR también proporciona una forma de replicar las imágenes en otros repositorios. Al especificar parámetros de replicación en la configuración del registro privado, puede replicar en otras regiones de su propio registro y en diferentes cuentas. Para obtener más información, consulte Configuración del registro privado en Amazon ECR.

Temas

- Permisos de IAM para la inserción de una imagen en un repositorio privado de Amazon ECR
- Inserción de una imagen de Docker en un repositorio privado de Amazon ECR
- Inserción de una imagen multiarquitectura a un repositorio privado de Amazon ECR
- Inserción de un gráfico de Helm en un repositorio privado de Amazon ECR

Permisos de IAM para la inserción de una imagen en un repositorio privado de Amazon ECR

Los usuarios necesitan contar con permisos de IAM para insertar imágenes en repositorios privados de Amazon ECR. Puede seguir la práctica recomendada de conceder privilegios mínimos y conceder acceso a un repositorio específico. También puede conceder acceso a todos los repositorios.

Un usuario debe autenticarse en cada registro de Amazon ECR al que desee insertar imágenes solicitando un token de autorización. Amazon ECR proporciona varias políticas AWS administradas para controlar el acceso de los usuarios en distintos niveles. Para obtener más información, consulte AWS políticas gestionadas para Amazon Elastic Container Registry.

También puede crear sus propias políticas de IAM. La siguiente política de IAM otorga los permisos necesarios para insertar una imagen en un repositorio específico. Para limitar los permisos de un repositorio específico, usa el nombre de recurso de Amazon (ARN) completo del repositorio.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:region:111122223333:repository/repository-name"
        },
        {
            "Effect": "Allow",
            "Action": "ecr:GetAuthorizationToken",
            "Resource": "*"
        }
    ]
}
```

La siguiente política de IAM otorga los permisos necesarios para insertar una imagen en todos los repositorios.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:GetAuthorizationToken",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage"
            ],
             "Resource": "arn:aws:ecr:region:111122223333:repository/*"
        }
    ]
}
```

Inserción de una imagen de Docker en un repositorio privado de Amazon ECR

Puede insertar las imágenes de contenedor a un repositorio de Amazon ECR con el comando docker push.

Amazon ECR también es compatible con la creación e inserción de listas de manifiesto de Docker que se utilizan para imágenes multiarquitectura. Para obtener información, consulte Inserción de una imagen multiarquitectura a un repositorio privado de Amazon ECR.

Inserción de una imagen de Docker en un repositorio de Amazon ECR

El repositorio de Amazon ECR debe existir antes de insertar la imagen. Para obtener más información, consulte the section called “Creación de un repositorio para almacenar imágenes”.

1. Autentique su cliente de Docker en el registro de Amazon ECR en el que va a insertar la imagen. Debe obtener tokens de autenticación para cada registro usado, cuya validez es de 12 horas. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

   Para autenticar Docker en un registro de Amazon ECR, ejecute el comando aws ecr get-login-password. Al pasar el token de autenticación al comando docker login, utilice el valor `AWS` para el nombre de usuario y especifique el URI del registro de Amazon ECR en el que desea autenticarse. Si se autentica en varios registros, deberá repetir el comando con cada registro.

   > **Important**
   >
   > Si recibe un error, instale o actualice a la versión más reciente de la AWS CLI. Para obtener más información, consulte Installing the AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

   ```
   aws ecr get-login-password --region region | docker login --username AWS --
   password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

2. Si el repositorio de imágenes no existe aún en el registro en el que lo va a insertar, créelo. Para obtener más información, consulte Creación de un repositorio privado de Amazon ECR para almacenar imágenes.
3. Identifique la imagen local que va a insertar. Ejecute el comando docker images para mostrar las imágenes de contenedor en el sistema.

```
docker images
```

Puede identificar una imagen con el *repository:tag* valor o el ID de la imagen en el resultado del comando resultante.

4. Etiquete su imagen con la combinación de nombres del registro, repositorio y etiqueta de imagen opcional de Amazon ECR que se van a usar. El formato del registro es *aws_account_id*.dkr.ecr.*region*.amazonaws.com. El nombre del repositorio debe coincidir con el repositorio que ha creado para su imagen. Si omite la etiqueta de imagen, se presupone que la etiqueta es `latest`.

   El siguiente ejemplo etiqueta una imagen local con el ID *e9ae3c220b23* como*aws_account_id*.dkr.ecr.*region*.amazonaws.com/my-repository:tag.

   ```
   docker tag e9ae3c220b23 aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag
   ```

5. Inserte la imagen mediante el comando docker push:

   ```
   docker push aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag
   ```

6. (Opcional) Aplique todas las demás etiquetas a su imagen e insértelas en Amazon ECR repitiendo Step 4 y Step 5.

Inserción de una imagen multiarquitectura a un repositorio privado de Amazon ECR

Puede insertar imágenes multiarquitectura en un repositorio de Amazon ECR mediante la creación e inserción de listas de manifiesto de Docker. Una lista de manifiesto es una lista de imágenes que se crea especificando uno o más nombres de imagen. En la mayoría de los casos, la lista de manifiesto se crea a partir de imágenes que cumplen la misma función, pero se utilizan para diferentes sistemas operativos o arquitecturas. La lista de manifiesto no es obligatoria. Para obtener más información, consulte manifiesto de docker.

Se puede extraer o hacer referencia a una lista de manifiesto en una definición de tarea de Amazon ECS o especificación de pod de Amazon EKS como otras imágenes de Amazon ECR.

Requisitos previos

- En la CLI de Docker, active las características experimentales. Para obtener información acerca de las características experimentales, consulte Experimental features en la documentación de Docker.
- El repositorio de Amazon ECR debe existir antes de insertar la imagen. Para obtener más información, consulte the section called “Creación de un repositorio para almacenar imágenes”.
- Las imágenes deben insertarse en el repositorio antes de crear el manifiesto de Docker. Para obtener información acerca de cómo insertar una imagen, consulte Inserción de una imagen de Docker en un repositorio privado de Amazon ECR.

Inserción de una imagen de Docker multiarquitectura a un repositorio de Amazon ECR

1. Autentique su cliente de Docker en el registro de Amazon ECR en el que va a insertar la imagen. Debe obtener tokens de autenticación para cada registro usado, cuya validez es de 12 horas. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

   Para autenticar Docker en un registro de Amazon ECR, ejecute el comando aws ecr get-login-password. Al pasar el token de autenticación al comando docker login, utilice el valor `AWS` para el nombre de usuario y especifique el URI del registro de Amazon ECR en el que desea autenticarse. Si se autentica en varios registros, deberá repetir el comando con cada registro.

   > **Important**
   >
   > Si recibe un error, instale o actualice a la versión más reciente de la AWS CLI. Para obtener más información, consulte Installing the AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

   ```
   aws ecr get-login-password --region region | docker login --username AWS --
   password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

2. Enumere las imágenes en su repositorio, confirmando las etiquetas de imagen.

   ```
   aws ecr describe-images --repository-name my-repository
   ```

3. Cree la lista de manifiestos de Docker. El comando `manifest create` verifica que las imágenes a las que se hace referencia ya estén en su repositorio y crea el manifiesto localmente.

```
docker manifest create aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:image_one_tag aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:image_two
```

4. (Opcional) Inspeccione la lista de manifiestos de Docker. Esto le permite confirmar el tamaño y la síntesis de cada manifiesto de imagen al que se hace referencia en la lista de manifiestos.

```
docker manifest inspect aws_account_id.dkr.ecr.region.amazonaws.com/my-repository
```

5. Inserte la lista de manifiesto de Docker en su repositorio de Amazon ECR.

```
docker manifest push aws_account_id.dkr.ecr.region.amazonaws.com/my-repository
```

Inserción de un gráfico de Helm en un repositorio privado de Amazon ECR

Puede insertar artefactos de iniciativa de contenedores abiertos (OCI) en un repositorio de Amazon ECR. Para ver un ejemplo de esta funcionalidad, siga los pasos siguientes a fin de insertar un gráfico de Helm en Amazon ECR.

Para obtener información acerca del uso de los gráficos de Helm alojados en Amazon ECR con Amazon EKS, consulte Instalación de un gráfico de Helm en un clúster de Amazon EKS.

Inserción de un gráfico de Helm a un repositorio de Amazon ECR

1. Instale la última versión del cliente Helm. Estos pasos se han escrito con la versión `3.8.2` de Helm. Para obtener más información, consulte Instalación de Helm.
2. Complete los pasos siguientes para crear un gráfico de Helm de prueba. Para obtener más información, consulte Documentación de Helm: Introducción.

   a. Cree un gráfico de Helm llamado `helm-test-chart` y borre el contenido del directorio `templates`.

   ```
   helm create helm-test-chart
   rm -rf ./helm-test-chart/templates/*
   ```

   b. Crea una `ConfigMap` en la `templates` carpeta.

```
cd helm-test-chart/templates
cat <<EOF > configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: helm-test-chart-configmap
data:
  myvalue: "Hello World"
EOF
```

3. Empaquete el gráfico. La salida incluirá el nombre de archivo del gráfico empaquetado que se utiliza al insertar el gráfico de Helm.

```
cd ../..
helm package helm-test-chart
```

Output

```
Successfully packaged chart and saved it to: /Users/username/helm-test-chart-0.1.0.tgz
```

4. Cree un repositorio para almacenar el gráfico de Helm. El nombre del repositorio debe coincidir con el que utiliza al crear el gráfico de Helm en el paso 2. Para obtener más información, consulte Creación de un repositorio privado de Amazon ECR para almacenar imágenes.

```
aws ecr create-repository \
     --repository-name helm-test-chart \
     --region us-west-2
```

5. Autentique el cliente Helm en el registro de Amazon ECR en el que tiene previsto insertar el gráfico de Helm. Debe obtener tokens de autenticación para cada registro usado, cuya validez es de 12 horas. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

```
aws ecr get-login-password \
     --region us-west-2 | helm registry login \
     --username AWS \
     --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

6. Inserte el gráfico de Helm con el comando helm push. La salida debe incluir el URI del repositorio de Amazon ECR y el resumen del Algoritmo hash seguro.

```
helm push helm-test-chart-0.1.0.tgz
 oci://aws_account_id.dkr.ecr.region.amazonaws.com/
```

7. Describa el gráfico de Helm.

```
aws ecr describe-images \
     --repository-name helm-test-chart \
     --region us-west-2
```

En el resultado, compruebe que el parámetro `artifactMediaType` indica el tipo de artefacto adecuado.

```
{
    "imageDetails": [
        {
            "registryId": "aws_account_id",
            "repositoryName": "helm-test-chart",
            "imageDigest":
 "sha256:dd8aebdda7df991a0ffe0b3d6c0cf315fd582cd26f9755a347a52adEXAMPLE",
            "imageTags": [
                "0.1.0"
            ],
            "imageSizeInBytes": 1620,
            "imagePushedAt": "2021-09-23T11:39:30-05:00",
            "imageManifestMediaType": "application/vnd.oci.image.manifest.v1+json",
            "artifactMediaType": "application/vnd.cncf.helm.config.v1+json"
        }
    ]
}
```

8. (Opcional) Para seguir pasos adicionales, instale el Helm `ConfigMap` y comience a utilizar Amazon EKS. Para obtener más información, consulte Instalación de un gráfico de Helm en un clúster de Amazon EKS.

Firma de una imagen almacenada en un repositorio privado de Amazon ECR

Amazon ECR se integra AWS Signer para proporcionarle una forma de firmar las imágenes de sus contenedores. Puede almacenar tanto las imágenes de contenedor como las firmas en sus repositorios privados.

Consideraciones

Se debe tener en cuenta lo siguiente al utilizar la firma de imágenes de Amazon ECR.

- Las firmas almacenadas en el repositorio se incluyen en la cuota de servicio para el número máximo de imágenes por repositorio. Para obtener más información, consulte Cuotas de servicio de Amazon ECR.
- Cuando hay artefactos de referencia en un repositorio, las políticas del ciclo de vida de Amazon ECR los limpiarán automáticamente en un plazo de 24 horas a partir de la eliminación de la imagen en cuestión.

Requisitos previos

Antes de comenzar, se deben cumplir los siguientes requisitos previos.

- Instalar y configurar la versión más reciente de la AWS CLI. Para obtener más información, consulte Instalación o actualización de la versión más reciente de la AWS CLI en la Guía del usuario de AWS Command Line Interface .
- Instale la CLI de Notation y el AWS Signer complemento de Notation. Para obtener más información, consulte Requisitos previos para firmar imágenes de contenedor en la Guía para desarrolladores de AWS Signer .
- Disponer de una imagen de contenedor guardada en un repositorio privado de Amazon ECR para firmarla. Para obtener más información, consulte Inserción de una imagen en un repositorio privado de Amazon ECR.

Configurar la autenticación para el cliente Notary

Antes de poder crear una firma mediante la CLI de Notation, debe configurar el cliente para que pueda autenticarse en Amazon ECR. Si tiene Docker instalado en el mismo host en el que instaló

el cliente de Notation, Notation reutilizará el mismo método de autenticación que utiliza para el cliente de Docker. Los comandos `login` y `logout` del Docker permitirán que los comandos `sign` y `verify` de Notation usen esas mismas credenciales y no tendrá que autenticar Notation por separado. Para obtener más información acerca de la configuración de su cliente de Notation para la autenticación, consulte Autenticar con registros compatibles con OCI en la documentación de Notary Project.

Si no utiliza Docker u otra herramienta que utilice credenciales de Docker, le recomendamos que utilice el Asistente de credenciales de Docker de Amazon ECR como almacén de credenciales. Para obtener más información sobre cómo instalar y configurar el asistente de credenciales de Amazon ECR, consulte el Asistente de credenciales de Amazon ECR Docker.

Firma de una imagen

Los siguientes pasos se pueden emplear para crear los recursos necesarios para firmar una imagen de contenedor y almacenar la firma en un repositorio privado de Amazon ECR. Notation firma las imágenes mediante el resumen.

Para firmar una imagen

1. Cree un perfil de AWS Signer firma mediante la plataforma de `Notation-OCI-SHA384-ECDSA` firma. Si lo desea, puede especificar un período de validez de la firma mediante el parámetro `--signature-validity-period`. Este valor puede especificarse utilizando `DAYS`, `MONTHS` o `YEARS`. Si no se especifica ningún periodo de validez, se utiliza el valor predeterminado de 135 meses.

   ```
   aws signer put-signing-profile --profile-name ecr_signing_profile --platform-id
    Notation-OCI-SHA384-ECDSA
   ```

   > **Note**
   >
   > El nombre del perfil de firma solo admite caracteres alfanuméricos y el guion bajo (_).

2. Autentique el cliente de Notation en el registro predeterminado. En el siguiente ejemplo, se utiliza AWS CLI para autenticar la CLI de Notation en un registro privado de Amazon ECR.

   ```
   aws ecr get-login-password --region region | notation login --username AWS --
   password-stdin 111122223333.dkr.ecr.region.amazonaws.com
   ```

3. Utilice la CLI de Notation para firmar la imagen y especifíquela mediante el nombre del repositorio y el resumen de SHA. Esto crea la firma y la envía al mismo repositorio privado de Amazon ECR donde se encuentre la imagen que se está firmando.

   En el siguiente ejemplo, se firma una imagen en el repositorio `curl` con el resumen de SHA `sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE`.

   ```
   notation
    sign 111122223333.dkr.ecr.region.amazonaws.com/
   curl@sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE --plugin
    "com.amazonaws.signer.notation.plugin" --id "arn:aws:signer:region:111122223333:/
   signing-profiles/ecrSigningProfileName"
   ```

Pasos a seguir a continuación

Luego de firmar la imagen de contenedor, puede verificar la firma localmente. Para obtener instrucciones acerca de cómo verificar una imagen, consulte Verificar una imagen localmente después de firmar en la Guía para desarrolladores de AWS Signer .

Eliminación de firmas y otros artefactos de un repositorio privado de Amazon ECR

Puede utilizar el cliente de ORAS para enumerar y eliminar firmas y otros artefactos de referencia de un repositorio privado de Amazon ECR. Eliminar firmas y otros artefactos de referencia es similar a eliminar una imagen (consulte Eliminación de una imagen en Amazon ECR). A continuación, se explica cómo enumerar artefactos y eliminar firmas:

Cómo administrar los artefactos de una imagen mediante la CLI de ORAS

1. Instalar y configurar el cliente de ORAS.

   Para obtener información acerca de la instalación y configuración del cliente de ORAS, consulte Instalación en la documentación de ORAS.
2. Para enumerar los artefactos disponibles para una imagen de Amazon ECR, utilice `oras discover`, seguido del nombre de la imagen:

   ```
   oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
   ```

La salida debe ser parecida a la siguiente:

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
### application/vnd.cncf.notary.signature
    ### sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
    ### sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

3. Para eliminar una firma mediante la CLI de ORAS, como se muestra en el ejemplo anterior, ejecute el siguiente comando:

```
oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

La salida debe ser parecida a la siguiente:

```
Are you sure you want to delete the manifest "111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and
 all tags associated with it? [y/N] y
```

4. Pulse y. Se debe eliminar el artefacto.

Cómo solucionar problemas de eliminación de artefactos

Si se produce un error al eliminar una firma, como se acaba de mostrar, aparecerá un resultado similar al siguiente.

```
Error response from registry: failed to delete 111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42:
unsupported: Requested image referenced by manifest list:
 [sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b]
```

Este error puede producirse al eliminar una imagen insertada antes del lanzamiento de OCI 1.1. Como se indica en el error, debe eliminar el manifiesto que hace referencia a la imagen antes de poder eliminar la imagen de la siguiente manera:

1. Para eliminar el manifiesto asociado a la firma que desea eliminar, escriba:

```
 oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b
```

   La salida debe ser parecida a la siguiente:

```
Are you sure you want to delete the manifest
 "sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b" and all
 tags associated with it? [y/N] y
```

2. Pulse y. Se debe eliminar el manifiesto.
3. Una vez eliminado el manifiesto, puede eliminar la firma:

```
 oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

   La salida debería ser similar a la siguiente. Pulse y.

```
Are you sure you want to delete the manifest
 "sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and all
 tags associated with it? [y/N] y
Deleted [registry] 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

4. Para comprobar que se ha eliminado la firma, escriba:

```
oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
```

   La salida debe ser parecida a la siguiente:

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
### application/vnd.cncf.notary.signature
    ### sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

Visualización de los detalles de una imagen en Amazon ECR

Luego de insertar una imagen en el repositorio, puede ver su información. Los detalles incluidos son los siguientes:

- URI de imagen
- Etiquetas de la imagen
- Tipo de medio del artefacto
- Tipo de manifiesto de imagen
- Estado de escaneo
- Tamaño de la imagen en MB
- Si la imagen se insertó en el repositorio
- Estado de la replicación

Visualización de los detalles de la imagen (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, elija la región que contiene el repositorio que incluye la imagen.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios), elija el repositorio que desea ver.
5. En la *repository_name* página Repositorios:, elija la imagen de la que desee ver los detalles.

Extracción de una imagen a su entorno local desde un repositorio privado de Amazon ECR

Si desea ejecutar una imagen de Docker que está disponible en Amazon ECR, puede extraerla a su entorno local con el comando docker pull. Puedes hacerlo desde tu registro predeterminado o desde un registro asociado a otra AWS cuenta.

Para utilizar una imagen de Amazon ECR en una definición de tarea de Amazon ECS, consulte Uso de imágenes de Amazon ECR con Amazon ECS.

> **Important**
>
> Amazon ECR requiere que los usuarios tengan permiso para realizar llamadas a la API `ecr:GetAuthorizationToken` a través de una política de IAM para que puedan autenticarse en un registro, así como insertar o extraer imágenes de cualquier repositorio de Amazon ECR. Amazon ECR proporciona varias políticas AWS administradas para controlar el acceso de los usuarios en distintos niveles. Para obtener información sobre las políticas AWS gestionadas de Amazon ECR, consulteAWS políticas gestionadas para Amazon Elastic Container Registry.

Extracción de una imagen de Docker de un repositorio de Amazon ECR

1. Autentique su cliente de Docker en el registro de Amazon ECR del que va a extraer la imagen. Debe obtener tokens de autenticación para cada registro usado, cuya validez es de 12 horas. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.
2. (Opcional) Identifique la imagen que va a extraer.
   - Puede mostrar los repositorios de un registro con el comando: aws ecr describe-repositories.

     ```
     aws ecr describe-repositories
     ```

     El registro de ejemplo anterior tiene un repositorio llamado `amazonlinux`.
   - Puede describir las imágenes de un repositorio con el comando: aws ecr describe-images.

     ```
     aws ecr describe-images --repository-name amazonlinux
     ```

El repositorio de ejemplo anterior tiene una imagen etiquetada como `latest` y `2016.09`, con el resumen de imagen `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807`.

3. Extraiga la imagen con el comando docker pull. El formato del nombre de imagen debe ser *registry*/*repository*[:*tag*] para extraer la imagen por etiqueta o *registry*/*repository*[@*digest*] para extraerla por resumen.

```
docker pull aws_account_id.dkr.ecr.us-west-2.amazonaws.com/amazonlinux:latest
```

> **Important**
>
> Si aparece un error *repository-url* `not found: does not exist or no pull access`, es posible que tenga que autenticar su cliente de Docker con Amazon ECR. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

Extracción de la imagen de contenedor Linux de Amazon

La imagen de contenedor de Amazon Linux se construye desde los mismos componentes de software que se incluyen en la AMI de Amazon Linux. La imagen de contenedor de Amazon Linux está disponible para su uso en cualquier entorno como imagen de base para cargas de trabajo de Docker. Si utiliza la AMI de Amazon Linux para las aplicaciones de Amazon EC2, puede contenerizar las aplicaciones con la imagen del contenedor de Amazon Linux.

Puede usar la imagen del contenedor de Amazon Linux en su entorno de desarrollo local y, a continuación, hacer que su aplicación AWS utilice Amazon ECS. Para obtener más información, consulte Uso de imágenes de Amazon ECR con Amazon ECS.

La imagen de contenedor de Amazon Linux está disponible en Amazon ECR Public y en Docker Hub. Para encontrar información acerca de la compatibilidad con la imagen de contenedor de Amazon Linux, consulte los foros para desarrolladores de AWS.

Extracción de la imagen de contenedor de Amazon Linux desde Amazon ECR Public

1. Autentique el cliente Docker en su registro de Amazon Linux Public. Los tokens de autenticación son válidos durante 12 horas. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

   > **Note**
   >
   > Los comandos ecr-public están disponibles en la AWS CLI a partir de la versión `1.18.1.187`, sin embargo, recomendamos utilizar la versión más reciente de la AWS CLI. Para obtener más información, consulte Installing the AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

   ```
   aws ecr-public get-login-password --region us-east-1 | docker login --username AWS
    --password-stdin public.ecr.aws
   ```

   La salida es la siguiente:

   ```
   Login succeeded
   ```

2. Extraiga la imagen de contenedor de Amazon Linux con el comando docker pull. Para ver la imagen de contenedor de Amazon Linux en la galería pública de Amazon ECR, consulte Galería pública de Amazon ECR: amazonlinux.

   ```
   docker pull public.ecr.aws/amazonlinux/amazonlinux:latest
   ```

3. (Opcional) Ejecute el contenedor localmente.

   ```
   docker run -it public.ecr.aws/amazonlinux/amazonlinux /bin/bash
   ```

Extracción de la imagen del contenedor de Amazon Linux de Docker Hub

1. Extraiga la imagen de contenedor de Amazon Linux con el comando docker pull.

   ```
   docker pull amazonlinux
   ```

2. (Opcional) Ejecute el contenedor localmente.

```
docker run -it amazonlinux:latest /bin/bash
```

Eliminación de una imagen en Amazon ECR

Si ha terminado de usar una imagen, puede eliminarla del repositorio. Si ha terminado de usar un repositorio, puede eliminar todo el repositorio y todas las imágenes que este contiene. Para obtener más información, consulte Eliminación de un repositorio privado en Amazon ECR.

Como alternativa a eliminar imágenes manualmente, puede crear políticas de ciclo de vida del repositorio que proporcionen más control sobre la administración del ciclo de vida de las imágenes en sus repositorios. Las políticas de ciclo de vida automatizan este proceso automáticamente. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.

> **Note**
>
> Si su repositorio tiene una mezcla de imágenes y algunas de ellas se insertaron antes de que Amazon ECR admitiera OCI v1.1, algunas firmas tendrán índices de imágenes o listas de manifiestos apuntando a ellas. Como resultado, al eliminar una imagen anterior a OCI v1.1, es posible que tenga que eliminar manualmente la lista de manifiestos que hace referencia a esa imagen y así eliminar el artefacto.

Eliminación de una imagen (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, seleccione la región que contiene la imagen que desea eliminar.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories, elija el repositorio del que contiene la imagen que desea eliminar.
5. En la ***repository_name*** página Repositorios:, seleccione la casilla situada a la izquierda de la imagen que desee eliminar y elija Eliminar.
6. En el cuadro de diálogo Eliminar imágenes, verifique las imágenes seleccionadas que deben eliminarse y elija Eliminar.

Eliminación de una imagen (AWS CLI)

1. Muestre una lista de las imágenes en el repositorio. Las imágenes etiquetadas tendrán un resumen de imagen y una lista de etiquetas asociadas. Las imágenes sin etiquetar solo tendrán un resumen de imagen.

```
aws ecr list-images \
     --repository-name my-repo
```

2. (Opcional) Elimine las etiquetas de la imagen que no desee especificando la etiqueta asociada a la imagen que desea eliminar. La imagen se elimina cuando se elimina la última etiqueta de esta.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageTag=tag1 imageTag=tag2
```

3. Elimine una imagen etiquetada o sin etiquetar especificando el resumen de imagen. Al eliminar una imagen haciendo referencia a su resumen, se elimina la imagen y todas sus etiquetas.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
```

   Para eliminar varias imágenes, puede especificar varias etiquetas de imagen o resúmenes de imagen en la solicitud.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
 imageDigest=sha256:f5t0e245ssffc302b13e25962d8f7a0bd304EXAMPLE
```

Reetiquetación de una imagen en Amazon ECR

Con las imágenes de Docker Image Manifest V2 Schema 2 puede usar la opción `--image-tag` del comando put-image para volver a etiquetar una imagen existente. Puede volver a etiquetar sin extraer o insertar la imagen con Docker. Para imágenes grandes, este proceso ahorra una cantidad considerable de ancho de banda de la red y del tiempo necesario para volver a etiquetar una imagen.

Para volver a etiquetar una imagen (AWS CLI)

Para volver a etiquetar una imagen con AWS CLI

1. Utilice el comando batch-get-image para obtener el manifiesto de la imagen para volver a etiquetarla y escribirla en un archivo. En este ejemplo, el manifiesto de una imagen con la etiqueta,*latest*, en el repositorio*amazonlinux*, se escribe en una variable de entorno denominada*MANIFEST*.

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --image-ids
 imageTag=latest --output text --query 'images[].imageManifest')
```

2. Use la opción `--image-tag` del comando put-image para colocar el manifiesto de imagen en Amazon ECR con una nueva etiqueta. En este ejemplo, la imagen está etiquetada como*2017.03*.

> **Note**
>
> Si la `--image-tag` opción no está disponible en su versión AWS CLI, actualice a la versión más reciente. Para obtener más información, consulte Installing the AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface .

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 --image-
manifest "$MANIFEST"
```

3. Verifique que la nueva etiqueta de imagen está asociada a la imagen. En la salida siguiente, la imagen tiene las etiquetas `latest` y `2017.03`.

```
aws ecr describe-images --repository-name amazonlinux
```

La salida es la siguiente:

```
{
    "imageDetails": [
        {
            "imageSizeInBytes": 98755613,
            "imageDigest":
 "sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a26EXAMPLE",
```

```
            "imageTags": [
                "latest",
                "2017.03"
            ],
            "registryId": "aws_account_id",
            "repositoryName": "amazonlinux",
            "imagePushedAt": 1499287667.0
        }
    ]
}
```

Para volver a etiquetar una imagen (AWS Tools for Windows PowerShell)

Para volver a etiquetar una imagen con AWS Tools for Windows PowerShell

1. Utilice Get-ECRImageBatch cmdlet para obtener la descripción de la imagen para volver a etiquetarla y escribirla en una variable de entorno. En este ejemplo, una imagen con la etiqueta,*latest*, en el repositorio,*amazonlinux*, se escribe en la variable de entorno,*$Image*.

   > **Note**
   >
   > Si no la tiene Get-ECRImageBatch cmdlet disponible en su sistema, consulte Configuración de la AWS Tools for Windows PowerShell en la Guía del Herramientas de AWS para PowerShell usuario.

   ```
   $Image = Get-ECRImageBatch -ImageId @{ imageTag="latest" } -
   RepositoryName amazonlinux
   ```

2. Escriba el manifiesto de la imagen en la variable de *$Manifest* entorno.

   ```
   $Manifest = $Image.Images[0].ImageManifest
   ```

3. Utilice la `-ImageTag` opción de Write-ECRImage cmdlet para colocar el manifiesto de imagen en Amazon ECR con una etiqueta nueva. En este ejemplo, la imagen está etiquetada como*2017.09*.

   ```
   Write-ECRImage -RepositoryName amazonlinux -ImageManifest $Manifest -
   ImageTag 2017.09
   ```

4. Verifique que la nueva etiqueta de imagen está asociada a la imagen. En la salida siguiente, la imagen tiene las etiquetas `latest` y `2017.09`.

   ```
   Get-ECRImage -RepositoryName amazonlinux
   ```

   La salida es la siguiente:

   ```
   ImageDigest                                                             ImageTag
   -----------                                                             --------
   sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 latest
   sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 2017.09
   ```

Cómo impedir que se sobrescriban las etiquetas de imagen en Amazon ECR

Puede activar la inmutabilidad de etiquetas en un repositorio y evitar así que se sobrescriban las etiquetas de imagen. Una vez que se active la inmutabilidad de etiquetas, se devuelve el error `ImageTagAlreadyExistsException` si se inserta una imagen con una etiqueta que ya existe en el repositorio. La inmutabilidad afecta a todas las etiquetas. No es posible hacer que algunas etiquetas sean inmutables mientras que otras no.

Puede usar las AWS CLI herramientas AWS Management Console y para configurar la mutabilidad de las etiquetas de imagen para un repositorio nuevo o para uno existente. Para crear un repositorio mediante los pasos de la consola, consulte Creación de un repositorio privado de Amazon ECR para almacenar imágenes.

Establecer la mutabilidad de las etiquetas de imagen (AWS Management Console)

Para establecer la mutabilidad de las etiquetas de imagen

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, elija la región que contiene el repositorio que desea editar.
3. En el panel de navegación, seleccione Repositorios en Registro privado.

   Si no ves Repositorios, selecciona Registro privado para expandir el menú y, a continuación, selecciona Repositorios.

4. En la página Repositorios privados, pulse el botón de radio situado antes del nombre del repositorio para el que desee establecer la configuración de mutabilidad de las etiquetas de imagen.
5. Selecciona Acciones y, a continuación, selecciona Repositorio en Editar.
6. Para la inmutabilidad de las etiquetas de imagen, elija una de las siguientes configuraciones de mutabilidad de etiquetas para el repositorio.

   - Mutable: elija esta opción si desea que se sobrescriban las etiquetas de imagen. Se recomienda para los repositorios que utilizan acciones de extracción de caché para garantizar que Amazon ECR pueda actualizar las imágenes almacenadas en caché. Además, para deshabilitar las actualizaciones de etiquetas para algunas etiquetas mutables, introduzca los nombres de las etiquetas o utilice caracteres comodín (*) para hacer coincidir varias etiquetas similares en el cuadro de texto de exclusión de etiquetas mutables.
   - Inmutable: elige esta opción si quieres evitar que las etiquetas de las imágenes se sobrescriban y se aplica a todas las etiquetas y exclusiones del repositorio al insertar una imagen con una etiqueta existente. Amazon ECR devuelve un `ImageTagAlreadyExistsException` si intentas insertar una imagen con una etiqueta existente. Además, para habilitar las actualizaciones de etiquetas para algunas etiquetas inmutables, introduzca los nombres de las etiquetas o utilice caracteres comodín (*) para hacer coincidir varias etiquetas similares en el cuadro de texto de exclusión de etiquetas inmutables.
7. En Image scan settings (Configuración de análisis de imágenes), si bien puede especificar la configuración de análisis en el nivel del repositorio para el análisis básico, la práctica recomendada es especificar la configuración de análisis en el nivel del registro privado. Especificar la configuración de análisis en el registro privado permite habilitar el análisis mejorado o el análisis básico, así como definir filtros para especificar qué repositorios se deben analizar. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.
8. Encryption settings (Configuración de cifrado) es un campo solo de visualización, ya que la configuración de cifrado de un repositorio no se puede cambiar una vez que se ha creado el repositorio.
9. Seleccione Save (Guardar) para actualizar la configuración del repositorio.

Establecer la mutabilidad de las etiquetas de imagen (AWS CLI)

Para crear un repositorio con etiquetas inmutables configuradas

Utilice uno de los siguientes comandos para crear un nuevo repositorio de imágenes con etiquetas inmutables configuradas.

- create-repository () con la posibilidad de mutar las etiquetas de imagen AWS CLI

```
aws ecr create-repository --repository-name name --image-tag-mutability IMMUTABLE --
region us-east-2
```

- create-repository (AWS CLI) con filtros de exclusión de la mutabilidad de las etiquetas de imagen

```
aws ecr create-repository --repository-name name --image-tag-
mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters
 filterType=WILDCARD,filter=filter-text --region us-east-2
```

- Nuevo- ECRRepository (AWS Tools for Windows PowerShell) con mutabilidad en las etiquetas de imagen

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE -Region us-
east-2 -Force
```

- Nuevo- ECRRepository (AWS Tools for Windows PowerShell) con filtros de exclusión de la mutabilidad de las etiquetas de imagen

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE_WITH_EXCLUSION
 -ImageTagMutabilityExclusionFilter @{FilterType=WILDCARD Filter=filter-text} -
Region us-east-2 -Force
```

Para actualizar la configuración de mutabilidad de las etiquetas de imagen en un repositorio

Utilice uno de los siguientes comandos para actualizar la configuración de mutabilidad de las etiquetas de imagen en un repositorio existente.

- put-image-tag-mutability(AWS CLI) con mutabilidad de etiquetas de imagen

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE --region us-east-2
```

- put-image-tag-mutability(AWS CLI) con filtros de exclusión de la mutabilidad de las etiquetas de imagen

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters
 filterType=WILDCARD,filter=latest --region us-east-2
```

- Escribe- ECRImage TagMutability (AWS Tools for Windows PowerShell) con mutabilidad en la etiqueta de imagen

```
Write-ECRImageTagMutability -RepositoryName name -ImageTagMutability IMMUTABLE -
Region us-east-2 -Force
```

- Write- ECRImage TagMutability (AWS Tools for Windows PowerShell) con filtros de exclusión de la mutabilidad de las etiquetas de imagen

```
Write-ECRImageTagMutability -RepositoryName name -
ImageTagMutability IMMUTABLE_WITH_EXCLUSION -ImageTagMutabilityExclusionFilter
 @{FilterType=WILDCARD Filter=latest}
```

Formatos de manifiesto de imagen de contenedor compatibles con Amazon ECR

Amazon ECR admite los siguientes formatos del manifiesto de imagen de contenedor:

- Docker Image Manifest V2 Schema 1 (usado con Docker versión 1.9 y anteriores)
- Docker Image Manifest V2 Schema 2 (usado con Docker versión 1.10 y posteriores)
- Especificaciones de la iniciativa de contenedores abiertos (OCI) (v1.0 y v1.1)

La compatibilidad con Docker Image Manifest V2 Schema 2 ofrece la siguiente funcionalidad:

- La capacidad de usar varias etiquetas para una imagen singular.
- Posibilidad de almacenar imágenes de contenedores Windows.

Conversión del manifiesto de imagen de Amazon ECR

Cuando inserta imágenes en Amazon ECR y las extrae desde él, el cliente del motor de contenedores (por ejemplo, Docker) se comunica con el registro para aceptar un formato de manifiesto que entiendan tanto el cliente como el registro para usarlo con la imagen.

Cuando inserta una imagen en Amazon ECR con Docker, versión 1.9 o anterior, el formato del manifiesto de imagen se almacena como Docker Image Manifest V2 Schema 1. Cuando inserta una imagen en Amazon ECR con Docker, versión 1.10 o una posterior, el formato del manifiesto de imagen se almacena como Docker Image Manifest V2 Schema 2.

Cuando extrae una imagen de Amazon ECR por etiqueta, este devuelve el formato del manifiesto de imagen almacenado en el repositorio. El formato se devuelve solo si el cliente entiende dicho formato. Si el cliente no entiende el formato del manifiesto de imagen almacenado, Amazon ECR convierte el manifiesto de imagen en un formato comprensible. Por ejemplo, si un cliente Docker 1.9 solicita un manifiesto de imagen que está almacenado como Docker Image Manifest V2 Schema 2, Amazon ECR devuelve el manifiesto en el formato Docker Image Manifest V2 Schema 1. En la tabla siguiente se describen las conversiones disponibles que admite Amazon ECR cuando se extrae una imagen por etiqueta:

Esquema solicitado por el cliente	Insertado en ECR como V2, esquema 1	Insertado en ECR como V2, esquema 2	Insertado en ECR como OCI
V2, esquema 1	No se requiere ninguna conversión	Convertido en V2, esquema 1	No hay ninguna conversión disponible
V2, esquema 2	No hay ninguna conversión disponible; el cliente utiliza V2, esquema 1	No se requiere ninguna conversión	Convertido en V2, esquema 2
OCI	No hay ninguna conversión disponible	Convertido en OCI	No se requiere ninguna conversión

Important

Si extrae una imagen por resumen, no hay ninguna conversión disponible. El cliente debe entender el formato del manifiesto de imagen almacenado en Amazon ECR. Si solicita

una imagen Docker Image Manifest V2 Schema 2 por resumen en un cliente Docker 1.9 o anterior, no se puede extraer la imagen. Para obtener más información, consulte Registry compatibility en la documentación de Docker.
En este ejemplo, si solicita la misma imagen por etiqueta, Amazon ECR convierte el manifiesto de imagen en un formato que el cliente pueda entender. La imagen se extrae correctamente.

Uso de imágenes de Amazon ECR con Amazon ECS

Puede utilizar los repositorios privados de Amazon ECR para alojar imágenes de contenedor y artefactos desde los que puedan realizar extracciones las tareas de Amazon ECS. Para que esto funcione, el agente de contenedores de Amazon ECS, o Fargate, debe tener permisos para crear `ecr:BatchGetImage`e`cr:GetDownloadUrlForLayer`, y. `ecr:GetAuthorizationToken` APIs

Permisos de IAM necesarios

En la siguiente tabla se muestra el rol de IAM que se debe utilizar para cada tipo de lanzamiento que proporciona los permisos necesarios para que las tareas puedan realizar extracciones desde un repositorio privado de Amazon ECR. Amazon ECS proporciona políticas de IAM administradas que incluyen los permisos necesarios.

Tipo de lanzamiento	rol de IAM	AWS política de IAM gestionada
Amazon ECS en EC2 instancias de Amazon	Utilice la función de IAM de la instancia de contenedor, que está asociada a la EC2 instancia de Amazon registrada en su clúster de Amazon ECS. Para obtener más información, consulte Rol de IAM de instancia de contenedor en la Guía para desarrolladores de Amazon Elastic Container Service.	`AmazonEC2ContainerServiceforEC2Role` Para obtener más información, consulte AmazonEC2ContainerServiceforEC2Roleen la Guía para desarrolladores de Amazon Elastic Container Service

Tipo de lanzamiento	rol de IAM	AWS política de IAM gestionada
Amazon ECS alojado en Fargate	Utilice el rol de IAM de ejecución de tareas al que haga referencia en la definición de la tarea de Amazon ECS. Para obtener más información, consulte Rol de IAM de ejecución de tareas en la Guía para desarrolladores de Amazon Elastic Container Service.	`AmazonECSTaskExecutionRolePolicy` Para obtener más información, consulte AmazonECSTaskExecutionRolePolicyen la Guía para desarrolladores de Amazon Elastic Container Service.
Amazon ECS en instancias externas	Utilice el rol de IAM de instancia de contenedor, que está asociado al servidor ubicado en las instalaciones o la máquina virtual (VM) registrados en el clúster de Amazon ECS. Para obtener más información, consulte Rol de IAM de instancia de contenedor de Amazon ECS en la Guía para desarrolladores de Amazon Elastic Container Service.	`AmazonEC2ContainerServiceforEC2Role` Para obtener más información, consulte AmazonEC2ContainerServiceforEC2Roleen la Guía para desarrolladores de Amazon Elastic Container Service.

Important

Las políticas de IAM AWS gestionadas contienen permisos adicionales que quizás no necesite para su uso. En tal caso, estos son los permisos mínimos necesarios para realizar extracciones desde un repositorio privado de Amazon ECR.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Especificación de una imagen de Amazon ECR en una definición de tarea de Amazon ECS

Cuando se crea una definición de tarea de Amazon ECS, se puede especificar una imagen de contenedor alojada en un repositorio privado de Amazon ECR. En la definición de tarea, asegúrese de utilizar la nomenclatura de `registry/repository:tag` completa para las imágenes de Amazon ECR. Por ejemplo, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-repository*`:`*latest*.

El siguiente fragmento de definición de tarea muestra la sintaxis que debería utilizar para especificar una imagen de contenedor alojada en Amazon ECR en la definición de tarea de Amazon ECS.

```
{
    "family": "task-definition-name",
    ...
    "containerDefinitions": [
        {
            "name": "container-name",
            "image": "aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:latest",
            ...
        }
    ],
    ...
}
```

Uso de imágenes de Amazon ECR con Amazon EKS

Puede utilizar sus imágenes de Amazon ECR con Amazon EKS.

Al hacer referencia a una imagen desde Amazon ECR, debe usar el nombre completo `registry/repository:tag` para la imagen. Por ejemplo, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-repository*`:`*latest*.

Permisos de IAM necesarios

Si tiene cargas de trabajo de Amazon EKS alojadas en nodos gestionados, nodos autogestionados o AWS Fargate consulte lo siguiente:

- Cargas de trabajo de Amazon EKS alojadas en nodos administrados o autoadministrados: el rol de IAM del nodo de trabajo de Amazon EKS (`NodeInstanceRole`) es obligatorio. El rol de IAM de nodo de trabajo de Amazon EKS debe contener los siguientes permisos de política de IAM para Amazon ECR.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Si utilizó `eksctl` o las AWS CloudFormation plantillas de Getting Started with Amazon EKS para crear sus grupos de clústeres y nodos de trabajo, estos permisos de IAM se aplican a su función de IAM de nodo de trabajo de forma predeterminada.

- Cargas de trabajo de Amazon EKS alojadas en AWS Fargate: utilice la función de ejecución de pods de Fargate, que permite a los pods extraer imágenes de repositorios privados de Amazon ECR. Para obtener más información, consulte Create a Fargate pod execution role Creación de un rol de ejecución de Fargate.

Instalación de un gráfico de Helm en un clúster de Amazon EKS

Los gráficos de Helm alojados en Amazon ECR se pueden instalar en sus clústeres de Amazon EKS.

Requisitos previos

- Instale la última versión del cliente Helm. Estos pasos se han escrito con la versión `3.9.0` de Helm. Para obtener más información, consulte Instalación de Helm.
- Tiene al menos una versión `1.23.9` o `2.6.3` del AWS CLI instalada en el equipo. Para obtener más información, consulte Instalación o actualización de la última versión del AWS CLI.
- Ha insertado un gráfico de Helm en el repositorio de Amazon ECR. Para obtener más información, consulte Inserción de un gráfico de Helm en un repositorio privado de Amazon ECR.
- Ha configurado `kubectl` para que funcione con Amazon EKS. Para obtener más información, consulte Creación de un elemento `kubeconfig` para Amazon EKS en la Guía del usuario de Amazon EKS. Si los siguientes comandos se realizan correctamente para su clúster, entonces tiene la configuración correcta.

```
kubectl get svc
```

Para instalar un gráfico de Helm en un clúster de Amazon EKS

1. Autentique su cliente Helm en el registro de Amazon ECR en el que esté alojado el gráfico de Helm. Debe obtener tokens de autenticación para cada registro usado, cuya validez es de 12 horas. Para obtener más información, consulte Autenticación de registro privado en Amazon ECR.

```
aws ecr get-login-password \
     --region us-west-2 | helm registry login \
     --username AWS \
     --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

2. Instale el gráfico. *helm-test-chart*Sustitúyalo por tu repositorio y por la etiqueta de tu *0.1.0* gráfico de Helm.

```
helm install ecr-chart-demo oci://aws_account_id.dkr.ecr.region.amazonaws.com/helm-test-chart --version 0.1.0
```

La salida debe ser parecida a la siguiente:

```
NAME: ecr-chart-demo
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

3. Verifique la instalación del gráfico.

```
helm list -n default
```

Ejemplo de salida:

```
NAME            NAMESPACE       REVISION        UPDATED
     STATUS          CHART                   APP VERSION
ecr-chart-demo  default         1               2022-06-01 15:56:40.128669157 +0000
 UTC deployed        helm-test-chart-0.1.0   1.16.0
```

4. (Opcional) Consulte el `ConfigMap` del gráfico de Helm instalado.

```
kubectl describe configmap helm-test-chart-configmap
```

5. Cuando haya terminado, puede quitar la versión del gráfico del clúster.

```
helm uninstall ecr-chart-demo
```

Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR

El escaneo de imágenes de Amazon ECR ayuda a identificar vulnerabilidades de software en las imágenes de contenedor. Se ofrecen los siguientes tipos de escaneo:

> **Important**
>
> Al cambiar entre las versiones de escaneo mejorado, escaneo básico y escaneo básico mejorado, los escaneos previamente establecidos dejarán de estar disponibles. Tendrá que volver a configurar los escaneos. Sin embargo, si regresa a la versión de escaneo anterior, los escaneos establecidos volverán a estar disponibles.

- Escaneo mejorado: Amazon ECR se integra con Amazon Inspector para ofrecer un escaneo automático y continuo de sus repositorios. Las imágenes de contenedor se analizan en busca de vulnerabilidades de los paquetes de idiomas de programación y sistemas operativos. A medida que aparecen nuevas vulnerabilidades, los resultados del análisis se actualizan y Amazon Inspector emite un evento EventBridge para notificártelo. El escaneo mejorado proporciona la siguiente información:
  - Los lenguajes de programación y los sistemas operativos empaquetan vulnerabilidades
  - Dos frecuencias de escaneo: escaneo pulsado y escaneo continuo
- Escaneo básico: Amazon ECR proporciona dos versiones del escaneo básico que utilizan la base de datos de vulnerabilidades y exposiciones comunes (CVEs):.
  - AWS Escaneo básico nativo: utiliza tecnología AWS nativa, que ahora es GA y se recomienda. Todos los registros de clientes nuevos están incluidos en esta versión mejorada de forma predeterminada.
  - Escaneo básico de Clair: utiliza el proyecto Clair de código abierto y está obsoleto.

  Con el escaneo básico, puede configurar los repositorios para el escaneo al insertar. También puede hacer escaneos manuales. Amazon ECR proporciona una lista de los resultados del escaneo. El escaneo básico proporciona la siguiente información:
  - Escaneos del sistema operativo
  - Dos frecuencias de escaneo: manual y escaneo por pulsación

> **Important**
>
> La nueva versión de Amazon ECR Basic Scanning no utiliza los `imageScanStatus` atributos `imageScanFindingsSummary` y de la respuesta de la `DescribeImages` API para devolver los resultados del escaneo. En su lugar, utilice la `DescribeImageScanFindings` API. Para obtener más información, consulte `DescribeImageScanFindings`.

Filtros para elegir qué repositorios se escanean en Amazon ECR

Al configurar el escaneo de imágenes para su registro privado, puede utilizar filtros para elegir qué repositorios se escanean.

Cuando se utiliza escaneo básico, puede especificar escaneo en filtros de inserción para determinar qué repositorios están configurados para realizar un escaneo de imagen cuando se inserten nuevas imágenes. Los repositorios que no coincidan con un análisis de escaneo básico en el filtro push se establecerán en la frecuencia de escaneo manual, lo que significa que para realizar un escaneo, debe activarlo manualmente.

Cuando se utiliza el escaneo mejorado, puede especificar filtros separados para escanear al insertar y escaneo continuo. Los repositorios que no coincidan con un filtro de escaneo mejorado tendrán deshabilitado el escaneo. Si utiliza la exploración mejorada y especifica filtros independientes para escanear al insertar y escaneo continuo donde varios filtros coinciden con el mismo repositorio, Amazon ECR aplica el filtro de escaneo continuo sobre el filtro de escaneo al insertar de ese repositorio.

Filtro de comodines

Cuando se especifica un filtro, un filtro sin comodín coincidirá con todos los nombres de repositorio que contengan el filtro. Un filtro con un comodín (*) coincide con cualquier nombre de repositorio en el que el comodín sustituya a cero o más caracteres del nombre del repositorio.

En la tabla siguiente se proporcionan ejemplos en los que los nombres de los repositorios se expresan en el eje horizontal y se especifican filtros de ejemplo en el eje vertical.

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod	Sí	Sí	Sí	Sí	Sí
*prod	Sí	Sí	No	No	No
prod*	Sí	No	Sí	No	Sí
prod	Sí	Sí	Sí	Sí	Sí
prod*repo	No	No	Sí	No	Sí

Escanee imágenes para detectar vulnerabilidades en los paquetes de lenguajes de programación y sistemas operativos en Amazon ECR

El escaneo mejorado de Amazon ECR es una integración con Amazon Inspector que proporciona análisis de vulnerabilidades para las imágenes de contenedor. Las imágenes de contenedor se analizan en busca de vulnerabilidades de los paquetes de idiomas de programación y sistemas operativos. Puede ver los hallazgos del escaneo tanto con Amazon ECR como con Amazon Inspector directamente. Para obtener información acerca de Amazon Inspector, consulte Escaneo de imágenes de contenedores con Amazon Inspector en la Guía del usuario de Amazon Inspector.

Con el escaneo mejorado, puede elegir qué repositorios están configurados para el escaneo automático y continuo y cuáles están configurados para el escaneo al insertar. Esto se hace configurando filtros de escaneo.

Consideraciones del escaneo mejorado

Antes de habilitar el escaneo mejorado de Amazon ECR, tenga en cuenta lo siguiente.

- El uso de esta característica no implica costos adicionales por parte de Amazon ECR; sin embargo, sí se generan cargos de Amazon Inspector por el escaneado de las imágenes. Esta función está disponible en las regiones en las que se admite Amazon Inspector. Para obtener más información, consulte:
  - Precios de Amazon Inspector: precios de Amazon Inspector.

  - Regiones compatibles con Amazon Inspector: regiones y puntos de conexión.
- El escaneo mejorado con Amazon ECR muestra cómo se utilizan las imágenes en Amazon EKS y Amazon ECS. Puede ver cuándo se usaron las imágenes por última vez e identificar cuántos clústeres utilizan cada imagen. Esta información le ayuda a priorizar la corrección de vulnerabilidades para las imágenes que se utilizan activamente. Puede determinar rápidamente qué clústeres podrían verse afectados por las vulnerabilidades descubiertas recientemente. Para obtener más información sobre cómo solicitar esta información y ver la respuesta, consulte `DescribeImageScanFindings`.
- Amazon Inspector admite el escaneo de sistemas operativos específicos. Para obtener una lista completa, consulte Sistemas operativos compatibles: escaneo de Amazon ECR en la Guía del usuario de Amazon Inspector.
- Amazon Inspector utiliza un rol de IAM vinculado a servicios, que proporciona los permisos necesarios para ofrecer un escaneo mejorado de los repositorios. Amazon Inspector crea automáticamente el rol de IAM vinculado a servicios cuando se enciende el escaneo mejorado para su registro privado. Para obtener más información, consulte Uso de roles vinculados a servicios de Amazon Inspector en la Guía del usuario de Amazon Inspector.
- Al activar por primera vez el escaneo mejorado para su registro privado, Amazon Inspector solo reconoce las imágenes enviadas a Amazon ECR en los últimos 14 días, según la marca de tiempo de inserción de la imagen. Las imágenes más antiguas tendrán el estado de escaneo `SCAN_ELIGIBILITY_EXPIRED`. Si desea que Amazon Inspector escanee estas imágenes, debe volver a subirlas a su repositorio.
- Cuando el registro privado de Amazon ECR tiene encendido el escaneo mejorado, todos los repositorios que coinciden con los filtros de escaneo se analizan únicamente mediante el escaneo mejorado. Cualquier repositorio que no coincida con un filtro tendrá una frecuencia de escaneo `Off`, pero no se escaneará. No se admiten los escaneos manuales mediante escaneo mejorado. Para obtener más información, consulte Filtros para elegir qué repositorios se escanean en Amazon ECR.
- Si especifica filtros independientes para escanear al insertar y escaneo continuo donde varios filtros coinciden con el mismo repositorio, Amazon ECR aplica el filtro de escaneo continuo sobre el filtro de escaneo al insertar de ese repositorio.
- Cuando se activa el escaneo mejorado, Amazon ECR envía un evento EventBridge cuando se cambia la frecuencia de escaneo de un repositorio. Amazon Inspector emite eventos EventBridge cuando se completa un escaneo inicial y cuando se crea, actualiza o cierra un hallazgo de escaneo de imágenes.

Cambio de la duración del escaneo mejorado de imágenes en Amazon Inspector

Tras habilitar el escaneo mejorado, Amazon ECR escanea continuamente las imágenes recién insertadas durante el tiempo configurado. De forma predeterminada, Amazon Inspector supervisa sus repositorios hasta que se eliminen las imágenes o se desactive el escaneo mejorado. Puede configurar tanto la duración de la fecha de inserción (hasta el ciclo de vida) como la duración de la nueva digitalización en la consola de Amazon Inspector para adaptarla a las necesidades de su entorno. Cuando transcurre la duración del escaneo de un repositorio, el estado del escaneo se muestra como. `SCAN_ELIGIBILITY_EXPIRED` Para obtener más información sobre cómo configurar los ajustes de duración de la redigitalización para Amazon ECR en Amazon Inspector, consulte Configuración de la duración de la redigitalización de Amazon ECR en la Guía del usuario de Amazon Inspector.

Para utilizar el escaneo mejorado en Amazon ECR, se requieren permisos de IAM

El escaneo mejorado de Amazon ECR requiere una función de IAM vinculada al servicio Amazon Inspector y que el director de IAM que habilite y utilice el escaneo mejorado tenga permisos para llamar al Inspector de Amazon necesario para escanear. APIs Amazon Inspector crea automáticamente el rol de IAM vinculado a servicios cuando se enciende el escaneo mejorado para su registro privado. Para obtener más información, consulte Uso de roles vinculados a servicios de Amazon Inspector en la Guía del usuario de Amazon Inspector.

La siguiente política de IAM concede los permisos necesarios para habilitar y utilizar el escaneo mejorado. Incluye el permiso necesario para que Amazon Inspector cree el rol de IAM vinculado a servicios, así como los permisos de la API de Amazon Inspector necesarios para encender y apagar el escaneo mejorado y recuperar los hallazgos del escaneo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "inspector2:Enable",
                "inspector2:Disable",
                "inspector2:ListFindings",
```

```
                "inspector2:ListAccountPermissions",
                "inspector2:ListCoverage"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "inspector2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Configuración de escaneos mejorados para imágenes en Amazon ECR

Configure el escaneo mejorado por región para su registro privado.

Asegúrese de contar con los permisos de IAM adecuados para configurar el escaneo mejorado. Para obtener información, consulte Para utilizar el escaneo mejorado en Amazon ECR, se requieren permisos de IAM.

AWS Management Console

Para encender el escaneo mejorado del registro privado

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, elija la región para la que desea establecer la configuración de escaneo.
3. En el panel de navegación, elija Registro privado y, a continuación, seleccione Configuración.
4. En la página Scanning configuration (Configuración de escaneo), para Scan type (Tipo de escaneo) elija Enhanced scanning (Escaneo mejorado).

   De forma predeterminada, cuando se selecciona Escaneo mejorado, todos los repositorios se escanean de forma continua.

5. Para elegir repositorios específicos y hacer que se escaneen de forma continua, desactive la casilla Escanear continuamente todos los repositorios y, a continuación, defina los filtros:

   > **Important**
   >
   > Los filtros sin comodines coincidirán con todos los nombres de repositorio que contengan el filtro. Los filtros con comodines (*) coinciden con un nombre de repositorio donde el comodín reemplaza cero o más caracteres del nombre. Para ver ejemplos del comportamiento de los filtros, consulte the section called “Filtro de comodines”.

   a. Introduzca un filtro basado en los nombres de los repositorios y, a continuación, seleccione Añadir filtro.

   b. Decida qué repositorios se escanearán cuando se inserte una imagen:

      - Para escanear todos los repositorios al insertar una imagen, seleccione Escanear todos los repositorios al insertar.
      - Para elegir repositorios específicos que se escanearán de forma automática al insertar, introduzca un filtro basado en los nombres de los repositorios y, a continuación, seleccione Añadir filtro.

6. Seleccione Save.
7. Repita estos pasos en cada región en la que desee encender el escaneo mejorado.

AWS CLI

Utilice el siguiente AWS CLI comando para activar el escaneo mejorado de su registro privado mediante el. AWS CLI Puede especificar filtros de escaneo mediante el objeto de `rules`.

- put-registry-scanning-configuration (AWS CLI)

  En el ejemplo siguiente se enciende el escaneo mejorado del registro privado. De forma predeterminada, cuando no se especifican `rules`, Amazon ECR establece la configuración de escaneo en escaneo continuo para todos los repositorios.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
```

```
    --region us-east-2
```

El siguiente ejemplo enciende un escaneo mejorado para su registro privado y especifica un filtro de escaneo. El filtro de escaneo del ejemplo enciende el escaneo continuo de todos los repositorios con `prod` en su nombre.

```
aws ecr put-registry-scanning-configuration \
    --scan-type ENHANCED \
    --rules '[{"repositoryFilters" : [{"filter":"prod","filterType" :
 "WILDCARD"}],"scanFrequency" : "CONTINUOUS_SCAN"}]' \
    --region us-east-2
```

El siguiente ejemplo enciende un escaneo mejorado para su registro privado y especifica varios filtros de escaneo. Los filtros de escaneo en el ejemplo enciende el escaneo continuo para todos los repositorios con `prod` en su nombre y enciende el escaneo al insertar solo para todos los demás repositorios.

```
aws ecr put-registry-scanning-configuration \
    --scan-type ENHANCED \
    --rules '[{"repositoryFilters" : [{"filter":"prod","filterType" :
 "WILDCARD"}],"scanFrequency" : "CONTINUOUS_SCAN"},{"repositoryFilters" :
 [{"filter":"*","filterType" : "WILDCARD"}],"scanFrequency" : "SCAN_ON_PUSH"}]' \
    --region us-west-2
```

EventBridge eventos enviados para un escaneo mejorado en Amazon ECR

Cuando se activa el escaneo mejorado, Amazon ECR envía un evento EventBridge cuando se cambia la frecuencia de escaneo de un repositorio. Amazon Inspector envía los eventos EventBridge cuando se completa un escaneo inicial y cuando se crea, actualiza o cierra un hallazgo de escaneo de imágenes.

Evento de cambio de frecuencia de escaneo de repositorio

Cuando el escaneo mejorado está encendido para su registro, Amazon ECR envía el siguiente evento cuando se produce un cambio con un recurso que tiene encendido el escaneo mejorado. Esto incluye la creación de nuevos repositorios, la frecuencia de escaneo de un repositorio que se está modificando o cuando se crean o eliminan imágenes en repositorios con el escaneo

mejorado encendido. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0abEXAMPLE",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
   {
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

Evento de un escaneo de imagen inicial (escaneo mejorado)

Cuando el escaneo mejorado está encendido para el registro, Amazon Inspector envía el siguiente evento cuando finaliza el escaneo de imágenes inicial. El parámetro `finding-severity-counts` solo devolverá un valor para un nivel de gravedad, si existe. Por ejemplo, si la imagen

no contiene resultados de nivel `CRITICAL`, no se devolverá ningún recuento crítico. Para obtener más información, consulte Escanee imágenes para detectar vulnerabilidades en los paquetes de lenguajes de programación y sistemas operativos en Amazon ECR.

Patrón de eventos:

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Scan"]
}
```

Ejemplo de salida:

```
{
    "version": "0",
    "id": "739c0d3c-4f02-85c7-5a88-94a9EXAMPLE",
    "detail-type": "Inspector2 Scan",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2021-12-03T18:03:16Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample"
    ],
    "detail": {
        "scan-status": "INITIAL_SCAN_COMPLETE",
        "repository-name": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/
amazon-ecs-sample",
        "finding-severity-counts": {
            "CRITICAL": 7,
            "HIGH": 61,
            "MEDIUM": 62,
            "TOTAL": 158
        },
        "image-digest":
 "sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
        "image-tags": [
            "latest"
        ]
    }
}
```

Evento para una actualización de búsqueda de escaneo de imagen (escaneo mejorado)

Cuando el escaneo mejorado está encendido para su registro, Amazon Inspector envía el siguiente evento cuando se crea, actualiza o cierra el hallazgo del escaneo de imágenes. Para obtener más información, consulte Escanee imágenes para detectar vulnerabilidades en los paquetes de lenguajes de programación y sistemas operativos en Amazon ECR.

Patrón de eventos:

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Finding"]
}
```

Ejemplo de salida:

```
{
    "version": "0",
    "id": "42dbea55-45ad-b2b4-87a8-afaEXAMPLE",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2021-12-03T18:02:30Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample/
sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77eEXAMPLE"
    ],
    "detail": {
        "awsAccountId": "123456789012",
        "description": "In libssh2 v1.9.0 and earlier versions, the SSH_MSG_DISCONNECT
 logic in packet.c has an integer overflow in a bounds check, enabling an attacker to
 specify an arbitrary (out-of-bounds) offset for a subsequent memory read. A crafted
 SSH server may be able to disclose sensitive information or cause a denial of service
 condition on the client system when a user connects to the server.",
        "findingArn": "arn:aws:inspector2:us-east-2:123456789012:finding/
be674aaddd0f75ac632055EXAMPLE",
        "firstObservedAt": "Dec 3, 2021, 6:02:30 PM",
        "inspectorScore": 6.5,
        "inspectorScoreDetails": {
            "adjustedCvss": {
                "adjustments": [],
                "cvssSource": "REDHAT_CVE",
```

```
                "score": 6.5,
                "scoreSource": "REDHAT_CVE",
                "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
                "version": "3.0"
            }
        },
        "lastObservedAt": "Dec 3, 2021, 6:02:30 PM",
        "packageVulnerabilityDetails": {
            "cvss": [
                {
                    "baseScore": 6.5,
                    "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
                    "source": "REDHAT_CVE",
                    "version": "3.0"
                },
                {
                    "baseScore": 5.8,
                    "scoringVector": "AV:N/AC:M/Au:N/C:P/I:N/A:P",
                    "source": "NVD",
                    "version": "2.0"
                },
                {
                    "baseScore": 8.1,
                    "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:H",
                    "source": "NVD",
                    "version": "3.1"
                }
            ],
            "referenceUrls": [
                "https://access.redhat.com/errata/RHSA-2020:3915"
            ],
            "source": "REDHAT_CVE",
            "sourceUrl": "https://access.redhat.com/security/cve/CVE-2019-17498",
            "vendorCreatedAt": "Oct 16, 2019, 12:00:00 AM",
            "vendorSeverity": "Moderate",
            "vulnerabilityId": "CVE-2019-17498",
            "vulnerablePackages": [
                {
                    "arch": "X86_64",
                    "epoch": 0,
                    "name": "libssh2",
                    "packageManager": "OS",
                    "release": "12.amzn2.2",
```

```
                    "sourceLayerHash":
 "sha256:72d97abdfae3b3c933ff41e39779cc72853d7bd9dc1e4800c5294dEXAMPLE",
                    "version": "1.4.3"
                }
            ]
        },
        "remediation": {
            "recommendation": {
                "text": "Update all packages in the vulnerable packages section to
 their latest versions."
            }
        },
        "resources": [
            {
                "details": {
                    "awsEcrContainerImage": {
                        "architecture": "amd64",
                        "imageHash":
 "sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
                        "imageTags": [
                            "latest"
                        ],
                        "platform": "AMAZON_LINUX_2",
                        "pushedAt": "Dec 3, 2021, 6:02:13 PM",
                        "lastInUseAt": "Dec 3, 2021, 6:02:13 PM",
                        "inUseCount": 1,
                        "registry": "123456789012",
                        "repositoryName": "amazon/amazon-ecs-sample"
                    }
                },
                "id": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-
sample/sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77EXAMPLE",
                "partition": "N/A",
                "region": "N/A",
                "type": "AWS_ECR_CONTAINER_IMAGE"
            }
        ],
        "severity": "MEDIUM",
        "status": "ACTIVE",
        "title": "CVE-2019-17498 - libssh2",
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Dec 3, 2021, 6:02:30 PM"
    }
```

```
}
```

Recuperación de los resultados de escaneos mejorados en Amazon ECR

Puede recuperar los resultados del último escaneo mejorado de imágenes completado y, a continuación, abrirlos en Amazon Inspector para ver más detalles. Las vulnerabilidades de software descubiertas se enumeran por gravedad según la base de datos de vulnerabilidades y exposiciones comunes (CVEs).

Para obtener detalles sobre la solución de problemas comunes al escanear imágenes, consulte Solución de problemas de escaneo de imágenes en Amazon ECR.

AWS Management Console

Siga los pasos siguientes para recuperar los resultados del escaneo de imágenes mediante la AWS Management Console.

Para recuperar los resultados del escaneo de imágenes

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la Región donde se encuentra su repositorio.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios) elija el repositorio que contiene la imagen para recuperar los resultados del escaneo.
5. Desde la página Imágenes, en la columna Etiqueta de imagen, seleccione la etiqueta de la imagen para recuperar los resultados del escaneo.
6. Para ver más detalles en la consola de Amazon Inspector, elija el nombre de la vulnerabilidad en la columna Nombre.

AWS CLI

Utilice el siguiente AWS CLI comando para recuperar los resultados del escaneo de imágenes mediante el AWS CLI. Puede especificar una imagen utilizando `imageTag` o `imageDigest`, ambos se pueden obtener mediante el comando list-images de la CLI.

- describe-image-scan-findings (AWS CLI)

  En el ejemplo siguiente se utiliza una etiqueta de imagen.

```
aws ecr describe-image-scan-findings \
     --repository-name name \
     --image-id imageTag=tag_name \
     --region us-east-2
```

En el ejemplo siguiente se utiliza un resumen de imágenes.

```
aws ecr describe-image-scan-findings \
     --repository-name name \
     --image-id imageDigest=sha256_hash \
     --region us-east-2
```

Escaneo de imágenes para detectar vulnerabilidades de sistema operativo en Amazon ECR

Amazon ECR ofrece dos versiones de análisis básico que utilizan la base de datos de vulnerabilidades y exposiciones comunes (CVEs):

- AWS escaneo básico nativo: utiliza tecnología AWS nativa, que ahora es GA y se recomienda. Este análisis básico mejorado está diseñado para ofrecer a los clientes mejores resultados de análisis y detección de vulnerabilidades en un amplio conjunto de sistemas operativos populares. Esto permite a los clientes reforzar aún más la seguridad de las imágenes de sus contenedores. Todos los registros de clientes nuevos están incluidos en esta versión mejorada de forma predeterminada.
- Análisis básico de Clair: la versión de análisis básico anterior, que utiliza el proyecto Clair de código abierto. Para obtener más información sobre Clair, consulte Clair en. GitHub

Tanto el escaneo AWS nativo como el básico de Clair se admiten en todas las regiones que figuran en los AWS Servicios por región, excepto en las que se agregaron después de septiembre de 2024. Debido a que la compatibilidad con Clair está obsoleta, Clair no será compatible en las nuevas regiones a medida que se vayan agregando y dejará de ser compatible en todas las regiones a partir del 1 de octubre de 2025.

Amazon ECR utiliza la gravedad de CVE del origen de distribución ascendente si está disponible. De lo contrario, se utiliza la puntuación del sistema de clasificación de vulnerabilidades comunes

(CVSS). La puntuación CVSS se puede utilizar para obtener la calificación de gravedad de vulnerabilidad de NVD. Para obtener más información, consulte NVD Vulnerability Severity Ratings.

Ambas versiones del escaneo básico de Amazon ECR admiten filtros para especificar qué repositorios se escanearán al insertar. Los repositorios que no coinciden con un filtro de escaneo al insertar se establecen en la frecuencia de escaneo manual, lo que significa que debe activar el escaneo manualmente. Una imagen se puede escanear una vez cada 24 horas. Dentro de esas 24 horas se incluye el escaneo al insertar inicial, si está configurado, y cualquier escaneo manual. Con el escaneo básico, puede digitalizar hasta 100 000 imágenes cada 24 horas en un registro determinado. El límite de 100 000 incluye tanto el escaneo inicial mediante inserción como el escaneo manual, tanto en Clair como en la versión mejorada del escaneo básico.

Para cada imagen se pueden recuperar los últimos resultados del escaneo de imágenes completados. Cuando se completa el escaneo de una imagen, Amazon ECR envía un evento a Amazon EventBridge. Para obtener más información, consulte Eventos de Amazon ECR y EventBridge.

Soporte del sistema operativo para el escaneo básico y el escaneo básico mejorado

Como práctica recomendada de seguridad y para una cobertura continua, recomendamos seguir utilizando versiones compatibles del sistema operativo. De acuerdo con la política del proveedor, los sistemas operativos discontinuos se han dejado de actualizar con parches y, en muchos casos, ya no se publican avisos de seguridad para ellos. A esto se suma que algunos proveedores eliminan los avisos de seguridad y las detecciones de sus fuentes cuando un sistema operativo afectado alcanza el final de la compatibilidad estándar. Luego de que una distribución pierde el soporte de su proveedor, es posible que Amazon ECR ya no admita escanearla en busca de vulnerabilidades. Todos los resultados que Amazon ECR genere con relación a un sistema operativo retirado tienen fines meramente informativos. A continuación se muestran los sistemas operativos y las versiones que son compatibles actualmente.

Sistema operativo	Versión	AWS nativo, básico	Clair básico
Alpine Linux (Alpine)	3.18	Sí	Sí
Alpine Linux (Alpine)	3,19	Sí	Sí

Sistema operativo	Versión	AWS nativo, básico	Clair básico
Alpine Linux (Alpine)	3.20	Sí	Sí
Alpine Linux (Alpine)	3.21	Sí	No
AlmaLinux	8	Sí	No
AlmaLinux	9	Sí	No
Amazon Linux (2AL2)	AL2	Sí	Sí
Amazon Linux 2023 (AL2023)	AL2023	Sí	Sí
Debian Server (Bookworm)	12	Sí	Sí
Debian Server (Bullseye)	11	Sí	Sí
Fedora	40	Sí	No
Fedora	41	Sí	No
OpenSUSE Leap	15.6	Sí	No
Oracle Linux (Oracle)	9	Sí	Sí
Oracle Linux (Oracle)	8	Sí	Sí

Sistema operativo	Versión	AWS nativo, básico	Clair básico
Photon OS	4	Sí	No
Photon OS	5	Sí	No
Red Hat Enterprise Linux (RHEL)	8	Sí	Sí
Red Hat Enterprise Linux (RHEL)	9	Sí	Sí
Rocky Linux	8	Sí	No
Rocky Linux	9	Sí	No
SUSE Linux Enterprise Server (SLES)	15.6	Sí	No
Ubuntu (Xenial)	16.04 (ESM)	Sí	Sí
Ubuntu (Bionic)	18.04 (ESM)	Sí	Sí
Ubuntu (Focal)	20.04 (LTS)	Sí	Sí
Ubuntu (Jammy)	22.04 (LTS)	Sí	Sí
Ubuntu (Noble Numbat)	24.04	Sí	No
Ubuntu (Oriole oracular)	24.10	Sí	No

Configuración de escaneos básicos para imágenes en Amazon ECR

De forma predeterminada, Amazon ECR activa el escaneo básico para todos los registros privados. Como resultado, a menos que haya cambiado la configuración de escaneo de su registro privado, no hay necesidad de activar el escaneo básico. El escaneo básico utiliza el proyecto Clair de código abierto.

Puede seguir los siguientes pasos para definir uno o más filtros de escaneo al insertar.

Para encender el escaneo básico para su registro privado

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/ private-registry/ repositories
2. En la barra de navegación, elija la región para la que desea establecer la configuración de escaneo.
3. En el panel de navegación, elija Private registry (Registro privado), Scanning (Escaneo).
4. En la página Scanning configuration (Configuración de escaneo) Para Scan type (Tipo de escaneo), elija Basic scanning (Escaneo básico).
5. De forma predeterminada, todos los repositorios están configurados para el escaneo manual. Opcionalmente, puede encender el escaneo push especificando Escaneo en filtros de inserción. Puede configurar el escaneo al insertar para todos los repositorios o para repositorios individuales. Para obtener más información, consulte Filtros para elegir qué repositorios se escanean en Amazon ECR.

Cambio al escaneo básico mejorado de imágenes en Amazon ECR

Amazon ECR ofrece capacidades mejoradas de escaneo de imágenes de contenedores mediante una versión mejorada del escaneo básico que utiliza tecnología AWS nativa. Esta función le ayuda a identificar las vulnerabilidades de software en las imágenes de sus contenedores. El siguiente procedimiento le ayuda a cambiar a esta versión mejorada del escaneo básico si utiliza una versión anterior del escaneo básico que utiliza `CLAIR` tecnología.

> **Important**
>
> En el caso de los nuevos usuarios, sus registros se configuran automáticamente para utilizar la tecnología de escaneo de `AWS_NATIVE` al momento de crearlos. No hay ninguna acción

que deba realizar. Amazon ECR no recomienda volver a la tecnología de escaneo anterior `CLAIR`.

AWS Management Console

Para activar el escaneo básico mejorado para su registro privado

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/ private-registry/repositories
2. En la barra de navegación, elija la región para la que desea establecer la configuración de escaneo.
3. En el panel de navegación, seleccione Registro privado, Características y configuración, Escaneo.
4. En la página de configuración del escaneo, seleccione Activar (recomendado) para seleccionar una versión mejorada del escaneo básico.
5. De forma predeterminada, todos los repositorios están configurados para el escaneo manual. Opcionalmente, puede encender el escaneo push especificando Escaneo en filtros de inserción. Puede configurar el escaneo al insertar para todos los repositorios o para repositorios individuales. Para obtener más información, consulte Filtros para elegir qué repositorios se escanean en Amazon ECR.

AWS CLI

Amazon ECR tiene habilitado el escaneo básico en todos los registros privados. Utilice los siguientes comandos para ver el tipo de escaneo básico actual y cambiarlo.

- Para recuperar la versión del tipo de escaneo básico que utiliza actualmente.

```
aws ecr get-account-setting --name BASIC_SCAN_TYPE_VERSION
```

El campo del nombre del parámetro es obligatorio. Si no proporciona el nombre, recibirá el siguiente mensaje de error:

```
aws: error: the following arguments are required: --name
```

Para cambiar la versión del tipo de escaneo básico de `CLAIR` a `AWS_NATIVE`. Una vez que se haya cambiado la versión básica del tipo de escaneo de `CLAIR` a `AWS_NATIVE`, no se recomienda volver a `CLAIR`.

```
aws ecr put-account-setting --name BASIC_SCAN_TYPE_VERSION --value value
```

Escaneo de imágenes manual para detectar vulnerabilidades de sistema operativo en Amazon ECR

Si sus repositorios no están configurados para escanear al insertar, puede iniciar los escaneos de imágenes manualmente. Una imagen se puede escanear una vez cada 24 horas. Dentro de esas 24 horas se incluye el escaneo al insertar inicial, si está configurado, y cualquier escaneo manual.

Para obtener detalles sobre la solución de problemas comunes al escanear imágenes, consulte Solución de problemas de escaneo de imágenes en Amazon ECR.

AWS Management Console

Siga los pasos siguientes para comenzar el escaneo manual de una imagen mediante la AWS Management Console.

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/ private-registry/repositories
2. En la barra de navegación, elija la región en la que va a crear el repositorio.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios), elija el repositorio del que contiene la imagen que desea escanear.
5. En la página Images (Imágenes) seleccione la imagen que desea escanear y, a continuación, elija Scan (Escanear).

AWS CLI

- start-image-scan (AWS CLI)

  En el ejemplo siguiente se utiliza una etiqueta de imagen.

```
aws ecr start-image-scan --repository-name name --image-id imageTag=tag_name --
region us-east-2
```

En el ejemplo siguiente se utiliza un resumen de imágenes.

```
aws ecr start-image-scan --repository-name name --image-id imageDigest=sha256_hash
 --region us-east-2
```

AWS Tools for Windows PowerShell

- ECRImageScanFinding AWS Tools for Windows PowerShell Obtenga- ()

  En el ejemplo siguiente se utiliza una etiqueta de imagen.

```
Start-ECRImageScan -RepositoryName name -ImageId_ImageTag tag_name -Region us-
east-2 -Force
```

  En el ejemplo siguiente se utiliza un resumen de imágenes.

```
Start-ECRImageScan -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2 -Force
```

Recuperación de los resultados de escaneos básicos en Amazon ECR

Puede recuperar los resultados del escaneo del último escaneo básico de imagen completado. Las vulnerabilidades de software que se descubrieron se enumeran por gravedad según la base de datos de vulnerabilidades y exposiciones comunes (CVEs).

Para obtener detalles sobre la solución de problemas comunes al escanear imágenes, consulte Solución de problemas de escaneo de imágenes en Amazon ECR.

AWS Management Console

Siga los pasos siguientes para recuperar los resultados del escaneo de imágenes mediante la AWS Management Console.

Para recuperar los resultados del escaneo de imágenes

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/ private-registry/repositories
2. En la barra de navegación, elija la región en la que va a crear el repositorio.
3. En el panel de navegación, elija Repositories (Repositorios).
4. En la página Repositories (Repositorios) elija el repositorio que contiene la imagen para recuperar los resultados del escaneo.
5. Desde la página Imágenes, en la columna Etiqueta de imagen, seleccione la etiqueta de la imagen para recuperar los resultados del escaneo.

AWS CLI

Utilice el siguiente AWS CLI comando para recuperar los resultados del escaneo de imágenes mediante el. AWS CLI Puede especificar una imagen utilizando `imageTag` o `imageDigest`, ambos se pueden obtener mediante el comando list-images de la CLI.

- describe-image-scan-findings (AWS CLI)

  En el ejemplo siguiente se utiliza una etiqueta de imagen.

  ```
  aws ecr describe-image-scan-findings --repository-name name --image-id
   imageTag=tag_name --region us-east-2
  ```

  En el ejemplo siguiente se utiliza un resumen de imágenes.

  ```
  aws ecr describe-image-scan-findings --repository-name name --image-id
   imageDigest=sha256_hash --region us-east-2
  ```

AWS Tools for Windows PowerShell

- Obtenga- ECRImage ScanFinding (AWS Tools for Windows PowerShell)

  En el ejemplo siguiente se utiliza una etiqueta de imagen.

  ```
  Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageTag tag_name -
  Region us-east-2
  ```

En el ejemplo siguiente se utiliza un resumen de imágenes.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2
```

Solución de problemas de escaneo de imágenes en Amazon ECR

A continuación se presentan errores comunes de escaneo de imágenes. Puede ver errores como este en la consola de Amazon ECR mostrando los detalles de la imagen o mediante la API o AWS CLI mediante la `DescribeImageScanFindings` API.

UnsupportedImageError

Puede obtener un error `UnsupportedImageError` al intentar escanear una imagen creada con un sistema operativo para el que Amazon ECR no admite el escaneo de imágenes básico. Amazon ECR admite el escaneo de vulnerabilidades de paquetes para las versiones principales de Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine y distribuciones de Linux RHEL. Una vez que una distribución pierde el soporte de su proveedor, es posible que Amazon ECR ya no admita escanearla en busca de vulnerabilidades. Amazon ECR no admite el escaneo de imágenes creadas a partir de la imagen temporal de Docker.

> **Important**
>
> Cuando se utiliza un escaneo mejorado, Amazon Inspector admite el escaneo de sistemas operativos y tipos de medios específicos. Para obtener una lista completa, consulte Sistemas operativos compatibles y tipos de medios en la Guía del usuario de Amazon Inspector.

Se devuelve un nivel de gravedad `UNDEFINED`

Es posible que reciba un resultado de escaneo que tenga un nivel de gravedad de `UNDEFINED`. Las siguientes son las causas comunes para esto:

- El origen de CVE no asignó prioridad a la vulnerabilidad.
- A la vulnerabilidad se le asignó una prioridad que Amazon ECR no reconocía.

Para determinar la gravedad y la descripción de una vulnerabilidad, puede ver las CVE directamente desde el origen.

Descripción del estado de análisis **SCAN_ELIGIBILITY_EXPIRED**

Cuando el análisis mejorado con Amazon Inspector está habilitado para su registro privado y está viendo sus vulnerabilidades de análisis, es posible que vea un estado de análisis de `SCAN_ELIGIBILITY_EXPIRED`. Estas pueden ser algunas de las causas:

- Al activar por primera vez el escaneo mejorado para su registro privado, Amazon Inspector solo reconoce las imágenes enviadas a Amazon ECR en los últimos 30 días, según la marca de tiempo de envío de las imágenes. Las imágenes más antiguas tendrán el estado de escaneo `SCAN_ELIGIBILITY_EXPIRED`. Si desea que Amazon Inspector escanee estas imágenes, debe volver a subirlas a su repositorio.
- Si el archivo de duración del reanálisis de ECR se cambia en la consola de Amazon Inspector y transcurre ese tiempo, el estado de análisis de la imagen cambia `ainactive` con un código de motivo de `expired`, y se programa el cierre de todos los hallazgos asociados a la imagen. Esto hace que la consola de Amazon ECR muestre el estado del análisis como `SCAN_ELIGIBILITY_EXPIRED`.

Sincronización de un registro principal con un registro privado de Amazon ECR

Mediante las reglas de caché de extracción, puede sincronizar el contenido de un registro principal con su registro privado de Amazon ECR.

Actualmente, Amazon ECR admite la creación de reglas de caché de extracción para los siguientes registros ascendentes:

- Amazon ECR Public, registro de imágenes de contenedores de Kubernetes y Quay (no requiere autenticación)
- Docker Hub, Microsoft Azure Container Registry, GitHub Container Registry y GitLab Container Registry (requiere autenticación con AWS Secrets Manager secreto)
- Amazon ECR (requiere autenticación con la función de AWS IAM)

En el GitLab caso de Container Registry, Amazon ECR solo admite la extracción GitLab de memoria caché con la oferta de software como servicio (SaaS). Para obtener más información sobre el uso GitLab de la oferta de SaaS, visite GitLab .com.

En el caso de los registros originales que requieren autenticación con secretos (como Docker Hub), debe almacenar sus credenciales en secreto. AWS Secrets Manager Puede utilizar la consola Amazon ECR para crear secretos de Secrets Manager para cada registro ascendente autenticado. Para obtener más información acerca de la creación de un secreto de Secrets Manager mediante la consola de Secrets Manager, consulte Almacenar en secreto las credenciales del repositorio principal AWS Secrets Manager.

En el caso de Amazon ECR, debe crear un rol de IAM si los registros de Amazon ECR ascendentes y descendentes pertenecen a una cuenta diferente. AWS Para obtener más información sobre cómo crear un rol de IAM, consulte Se requieren políticas de IAM para transferir la memoria caché de ECR a ECR entre cuentas.

Después de crear una regla de extracción de caché para el registro principal, extraiga una imagen de ese registro principal con el URI de registro privado de Amazon ECR. A continuación, Amazon ECR crea un repositorio y guarda en caché la imagen en su registro privado. Para las solicitudes de extracción posteriores de la imagen en caché con una etiqueta determinada, Amazon ECR comprueba el registro principal para ver si hay una nueva versión de la imagen con esa etiqueta específica e intenta actualizar la imagen en su registro privado al menos una vez cada 24 horas.

Plantillas de creación de repositorios

Amazon ECR ha agregado soporte para las plantillas de creación de repositorios, lo que le permite especificar las configuraciones iniciales de los nuevos repositorios creados por Amazon ECR en su nombre mediante reglas de caché de extracción. Cada plantilla contiene un prefijo de espacio de nombres de repositorio que se utiliza para hacer coincidir los nuevos repositorios con una plantilla específica. Las plantillas pueden especificar la configuración de todos los ajustes del repositorio, incluidas las políticas de acceso basadas en los recursos, la inmutabilidad de las etiquetas, el cifrado y las políticas de ciclo de vida. La configuración de una plantilla de creación de repositorios solo se aplica durante la creación del repositorio y no tiene ningún efecto en los repositorios existentes ni en los repositorios creados mediante cualquier otro método. Para obtener más información, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.

Consideraciones para utilizar las reglas de caché de extracción

Tenga en cuenta lo siguiente al utilizar las reglas de caché de extracción de Amazon ECR.

- La creación de reglas de caché de extracción no se admite en las siguientes regiones.
  - China (Pekín) (`cn-north-1`)
  - China (Ningxia) (`cn-northwest-1`)
  - AWS GovCloud (Este de EE. UU.) () `us-gov-east-1`
  - AWS GovCloud (EEUU-Oeste) () `us-gov-west-1`
- AWS Lambda no admite la extracción de imágenes de contenedores de Amazon ECR mediante una regla de extracción de caché.
- Al extraer imágenes mediante caché de extracción, los puntos de conexión de servicio de FIPS de Amazon ECR no se admiten la primera vez que se extrae una imagen. Sin embargo, el uso de los puntos de conexión de servicio de FIPS de Amazon ECR funciona en extracciones posteriores.
- Cuando se extrae una imagen en caché a través del URI del registro privado de Amazon ECR, las extracciones de imágenes se inician mediante AWS direcciones IP. Esto garantiza que la extracción de imágenes no se tenga en cuenta en ninguna cuota de tasa de extracción implementada por el registro ascendente.
- Cuando una imagen almacenada en caché se extrae a través del URI del registro privado de Amazon ECR, Amazon ECR comprueba el repositorio ascendente al menos una vez cada 24 horas para verificar si la imagen almacenada en caché es la última versión. Si hay una imagen

más reciente en el registro original, Amazon ECR intenta actualizar la imagen en caché. Este temporizador se basa en la última extracción de la imagen almacenada en caché.

- Si Amazon ECR no puede actualizar la imagen desde el registro original por algún motivo y se extrae la imagen, se seguirá extrayendo la última imagen en caché.
- Al crear el secreto de Secrets Manager que contiene las credenciales de registro anteriores, el nombre del secreto debe usar el `ecr-pullthroughcache/` prefijo. El secreto también debe estar en la misma cuenta y región en las que se creó la regla de caché de extracción.
- Cuando se extrae una imagen multiarquitectura mediante una regla de caché de extracción, la lista de manifiestos y cada imagen a la que se hace referencia en la lista de manifiestos se extraen al repositorio de Amazon ECR. Si solo desea extraer una arquitectura específica, puede extraer la imagen mediante el resumen de imagen o la etiqueta asociada a la arquitectura en lugar de la etiqueta asociada a la lista de manifiestos.
- Amazon ECR utiliza un rol de IAM vinculada al servicio, que proporciona los permisos necesarios para que Amazon ECR cree el repositorio, recupere el valor secreto de Secrets Manager para la autenticación y envíe la imagen almacenada en caché en su nombre. El rol de IAM vinculado a servicios se crea automáticamente cuando se crea una regla de caché de extracción. Para obtener más información, consulte Rol vinculado a servicios de Amazon ECR para la caché de extracción.
- De forma predeterminada, la entidad principal de IAM que extrae la imagen almacenada en caché tiene los permisos otorgados a través de su política de IAM. Puede utilizar la política de permisos de registro privado de Amazon ECR para ampliar los permisos de una entidad de IAM. Para obtener más información, consulte Uso de permisos de registro.
- Los repositorios de Amazon ECR creados mediante el flujo de trabajo de caché de extracción se tratan como cualquier otro repositorio de Amazon ECR. Se admiten todas las características del repositorio, como la replicación y el escaneo de imágenes.
- Cuando Amazon ECR crea un repositorio nuevo en su nombre mediante una acción de caché de extracción, se aplican las siguientes configuraciones predeterminadas al repositorio, a menos que haya una plantilla de creación de repositorios coincidente. Puede utilizar una plantilla de creación de repositorios para definir la configuración que se aplicará a los repositorios creados por Amazon ECR en su nombre. Para obtener más información, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.
  - Inmutabilidad de etiquetas: la inmutabilidad de etiquetas especifica si las etiquetas de imagen se pueden sobrescribir. De forma predeterminada, las etiquetas de imagen son mutables (se pueden sobrescribir). Puede modificar el comportamiento de las etiquetas configurando los filtros de exclusión de etiquetas en el cuadro de texto de exclusión de etiquetas mutables cuando se

selecciona Mutable o en el cuadro de texto de exclusión de etiquetas inmutables cuando se selecciona Inmutable.

- Cifrado: se utiliza el cifrado predeterminado. `AES256`
- Permisos de repositorio: omitidos, no se aplica ninguna política de permisos de repositorio.
- Política de ciclo de vida: si se omite, no se aplica ninguna política de ciclo de vida.
- Etiquetas de recursos: se omiten, no se aplica ninguna etiqueta de recursos.

- Al activar la inmutabilidad de las etiquetas de imagen en los repositorios mediante una regla de caché de extracción impedirá que Amazon ECR actualice las imágenes que utilicen la misma etiqueta.
- Cuando se extrae una imagen mediante la regla de caché de extracción por primera vez, es posible que se requiera una ruta a Internet. Hay ciertas circunstancias en las que se requiere una ruta a Internet, por lo que es mejor configurar una ruta para evitar errores. Por lo tanto, si ha configurado Amazon ECR para que utilice una interfaz que AWS PrivateLink utilice un punto de enlace de VPC, debe asegurarse de que la primera extracción tenga una ruta a Internet. Una forma de hacerlo es crear una subred pública en la misma VPC, con una puerta de enlace de Internet, y luego enrutar todo el tráfico saliente a Internet desde su subred privada a la subred pública. Esto no será necesario la próxima vez que realice una extracción de imágenes mediante la regla de caché de extracción. Para obtener más información, consulte Opciones de enrutamiento de ejemplo en la guía de usuarios de Amazon Virtual Private Cloud.

Permisos de IAM necesarios para sincronizar un registro principal con un registro privado de Amazon ECR

Además de los permisos de la API de Amazon ECR necesarios para autenticarse en un registro privado y para insertar y extraer imágenes, se necesitan los siguientes permisos adicionales para utilizar reglas de caché de extracción.

- `ecr:CreatePullThroughCacheRule`: otorga permiso para crear una regla de caché de extracción. Este permiso debe otorgarse a través de una política de IAM basada en identidad.
- `ecr:BatchImportUpstreamImage`: otorga permiso para recuperar la imagen externa e importarla a su registro privado. Este permiso se puede otorgar utilizando la política de permisos de registro privado, una política de IAM basada en identidad o mediante la política de permisos de repositorio basada en recursos. Para obtener más información sobre el uso de permisos de repositorio, consulte Políticas de repositorios privados en Amazon ECR.

- `ecr:CreateRepository`: otorga permiso para crear un repositorio en un registro privado. Este permiso es necesario si el repositorio donde se conservan las imágenes almacenadas en caché no existe todavía. Este permiso se puede otorgar mediante una política de IAM basada en identidad o bien mediante la política de permisos de registro privado.

Uso de permisos de registro

Los permisos de registro privado de Amazon ECR se pueden utilizar para abarcar los permisos de entidades de IAM individuales a fin de utilizar caché de extracción. Si una entidad de IAM tiene más permisos otorgados por una política de IAM de los que la política de permisos de registro concede, prevalece la política de IAM. Por ejemplo, si un usuario tiene concedidos permisos de `ecr:*`, no se necesitan permisos adicionales en el nivel del registro.

Para crear una política de permisos de registro privado (AWS Management Console):

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que configurar su declaración de permisos de registro privado.
3. En el panel de navegación, elija Private registry (Registro privado), Registry permissions (Permisos de registro).
4. En la página Registry permissions (Permisos de registro), elija Generate statement (Generar declaración) .
5. Para cada declaración de política de permisos de caché de extracción que desee crear, haga lo siguiente.

    a. Para Policy type (Tipo de política), elija Pull through cache policy (Política de caché de extracción).

    b. Para Statement id (ID de declaración), proporcione un nombre para la política de declaración de caché de extracción.

    c. Para AM entities (Entidades de IAM), especifique los usuarios, grupos o roles que se deben incluir en la política.

    d. Para Repository namespace (Espacio de nombres de repositorio), seleccione la regla de caché de extracción a la que asociar la política.

    e. Para Repository names (Nombres de repositorio), especifique el nombre base del repositorio para el que se va a aplicar la regla. Por ejemplo, si desea especificar el

repositorio de Amazon Linux en Amazon ECR Public, el nombre del repositorio sería `amazonlinux`.

Para crear una política de permisos de registro privado (AWS CLI):

Utilice el siguiente AWS CLI comando para especificar los permisos de registro privado mediante el AWS CLI.

1. Cree un archivo local denominado `ptc-registry-policy.json` con el contenido de la política de registro. En el siguiente ejemplo se concede el permiso `ecr-pull-through-cache-user` para crear un repositorio y extraer una imagen de la galería pública de Amazon ECR, que es el origen ascendente asociado a la regla de caché de extracción creada previamente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PullThroughCacheFromReadOnlyRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/ecr-pull-through-cache-user"
      },
      "Action": [
        "ecr:CreateRepository",
        "ecr:BatchImportUpstreamImage"
      ],
      "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/ecr-public/*"
    }
  ]
}
```

> **Important**
>
> El permiso `ecr-CreateRepository` solo es necesario si el repositorio donde se conservan las imágenes almacenadas en caché no existe todavía. Por ejemplo, si la acción de creación del repositorio y las acciones de extracción de imágenes las realizan entidades principales de IAM independientes, como un administrador y un desarrollador.

2. Utilice el put-registry-policycomando para establecer la política de registro.

```
aws ecr put-registry-policy \
     --policy-text file://ptc-registry.policy.json
```

Pasos a seguir a continuación

Una vez que esté listo para empezar a utilizar las reglas de caché de extracción, siga estos pasos.

- Crear una regla de caché de extracción. Para obtener más información, consulte Creación de una regla de caché de extracción en Amazon ECR.
- Cree una plantilla de creación de repositorios. Puede utilizar una plantilla de creación de repositorios para definir la configuración que se aplicará a los repositorios creados por Amazon ECR en su nombre durante una acción de caché de extracción. Para obtener más información, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.

Configuración de los permisos de ECR multicuenta a ECR PTC

La función de caché de extracción de Amazon ECR a Amazon ECR (ECR a ECR) permite la sincronización automática de imágenes entre regiones, AWS cuentas o ambas. Con ECR a ECR PTC, puede insertar imágenes en su registro principal de Amazon ECR y configurar una regla de extracción de caché para almacenar en caché las imágenes en los registros de Amazon ECR descendentes.

Se requieren políticas de IAM para transferir la memoria caché de ECR a ECR entre cuentas

Para almacenar en caché imágenes entre los registros de Amazon ECR de distintas AWS cuentas, cree un rol de IAM en la cuenta descendente y configure las políticas de esta sección para proporcionar los siguientes permisos:

- Amazon ECR necesita permisos para extraer imágenes del registro original de Amazon ECR en su nombre. Para conceder estos permisos, cree una función de IAM y, a continuación, la especifique en su regla de extracción de memoria caché.
- El propietario del registro original también debe conceder al propietario del registro de la caché los permisos necesarios para incluir las imágenes en las políticas de recursos.

Políticas

- Crear una función de IAM para definir los permisos de extracción de memoria caché
- Crear una política de confianza para el rol de IAM
- Creación de una política de recursos en el registro original de Amazon ECR

Crear una función de IAM para definir los permisos de extracción de memoria caché

El siguiente ejemplo muestra una política de permisos que concede a un rol de IAM permiso para extraer imágenes del registro principal de Amazon ECR en su nombre. Cuando Amazon ECR asume la función, recibe los permisos especificados en esta política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken",
                "ecr:BatchImportUpstreamImage",
                "ecr:BatchGetImage",
                "ecr:GetImageCopyStatus",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

Crear una política de confianza para el rol de IAM

El siguiente ejemplo muestra una política de confianza que identifica a Amazon ECR pull through cache como el principal de AWS servicio que puede asumir la función.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "pullthroughcache.ecr.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Creación de una política de recursos en el registro original de Amazon ECR

El propietario del registro principal de Amazon ECR también debe añadir una política de registro o una política de repositorio para conceder al propietario del registro descendente los permisos necesarios para realizar las siguientes acciones.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::444455556666:root"
    },
    "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchImportUpstreamImage",
        "ecr:GetImageCopyStatus"
    ],
    "Resource": "arn:aws:ecr:region:111122223333:repository/*"
}
```

Creación de una regla de caché de extracción en Amazon ECR

Para cada registro principal que contenga imágenes que desee almacenar en caché en su registro privado de Amazon ECR, deberá crear una regla de caché de extracción.

En el caso de los registros anteriores que requieren autenticación con secretos, debe almacenar las credenciales en un secreto de Secrets Manager. Puede usar un secreto existente de o crear uno nuevo. Puede crear el secreto de Secrets Manager en la consola de Amazon ECR o en la de Secrets Manager. Para crear el secreto de Secrets Manager en la consola de Secrets Manager en lugar de

la de Amazon ECR, consulte Almacenar en secreto las credenciales del repositorio principal AWS Secrets Manager.

Requisitos previos

- Compruebe que tiene los permisos de IAM necesarios para crear reglas de caché de extracción. Para obtener información, consulte Permisos de IAM necesarios para sincronizar un registro principal con un registro privado de Amazon ECR.
- Para los registros anteriores que requieren autenticación con secretos: si desea utilizar un secreto existente, compruebe que el secreto de Secrets Manager cumpla los siguientes requisitos:
  - El nombre del secreto comienza con `ecr-pullthroughcache/`. La AWS Management Console solo muestra los secretos de Secrets Manager con el prefijo `ecr-pullthroughcache/`.
  - La cuenta y la región en las que se encuentra el secreto deben coincidir con la cuenta y la región en las que se encuentra la regla de caché de extracción.

Para crear una regla de caché de extracción (AWS Management Console)

Los siguientes pasos muestran cómo crear una regla de caché de extracción y un secreto de Secrets Manager mediante la consola de Amazon ECR. Para crear un secreto desde la consola de Secrets Manager, consulte Almacenar en secreto las credenciales del repositorio principal AWS Secrets Manager.

Para Amazon ECR Public, Kubernetes Container Registry o Quay

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. Desde la barra de navegación, elija la región en la que desea configurar su registro privado.
3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).
4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).
5. En la página del Paso 1: Especificar un origen, en Registro, elija Amazon ECR Public, Kubernetes o Quay de la lista de registros principales y, a continuación, elija Siguiente.
6. En la página del Paso 2: Especificar un destino, en Prefijo del repositorio de Amazon ECR, especifique el prefijo del espacio de nombres del repositorio que se utilizará al almacenar en caché las imágenes extraídas del registro público de origen y, a continuación, seleccione

Siguiente. De forma predeterminada, se rellena un espacio de nombres pero también se puede especificar un espacio de nombres personalizado.

7. En la página del Paso 3: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.
8. Repita el paso anterior para cada caché de extracción que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para Docker Hub

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. Desde la barra de navegación, elija la región en la que desea configurar su registro privado.
3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).
4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).
5. En la página del Paso 1: Especificar un origen, en Registro, elija Docker Hub, Siguiente.
6. En la página del Paso 2: Configurar la autenticación, en Credenciales principales, debe almacenar las credenciales de autenticación de Docker Hub en un secreto de AWS Secrets Manager . Puede especificar un secreto existente o utilizar la consola Amazon ECR para crear uno nuevo.

   a. Para usar un secreto existente, elija Usar un AWS secreto existente. En Nombre secreto, usa el menú desplegable para seleccionar tu secreto existente y, a continuación, selecciona Siguiente.

   > **Note**
   >
   > AWS Management Console Solo muestra los secretos de Secrets Manager con nombres que usan el `ecr-pullthroughcache/` prefijo. El secreto también debe estar en la misma cuenta y región en las que se creó la regla de caché de extracción.

   b. Para crear un secreto nuevo, selecciona Crear un AWS secreto, haz lo siguiente y, a continuación, selecciona Siguiente.

i. En Nombre secreto, especifique un nombre descriptivo para el secreto. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode.

ii. En correo de Docker Hub, especifique su correo de Docker Hub.

iii. Para el token de acceso a Docker Hub, especifique su token de acceso a Docker Hub. Para obtener más información sobre cómo crear un token de acceso de Docker Hub, consulta Crear y administrar los tokens de acceso en la documentación de Docker.

7. En la página del Paso 3: Especificar un destino, para Prefijo del repositorio de Amazon ECR, especifique el prefijo del espacio de nombres del repositorio que se utilizará al almacenar en caché las imágenes extraídas del registro público de origen y, a continuación, seleccione Siguiente.

   De forma predeterminada, se rellena un espacio de nombres pero también se puede especificar un espacio de nombres personalizado.

8. En la página del Paso 4: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.

9. Repita el paso anterior para cada caché de extracción que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para GitHub Container Registry

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.

2. Desde la barra de navegación, elija la región en la que desea configurar su registro privado.

3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).

4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).

5. En el paso 1: especifique una página de origen, en Registry, elija GitHub Container Registry y, a continuación.

6. En la página Paso 2: Configurar la autenticación, en el caso de las credenciales de Upstream, debe guardar en AWS Secrets Manager secreto las credenciales de autenticación de GitHub Container Registry. Puede especificar un secreto existente o utilizar la consola Amazon ECR para crear uno nuevo.

a. Para usar un secreto existente, selecciona Usar un AWS secreto existente. En Nombre secreto, usa el menú desplegable para seleccionar tu secreto existente y, a continuación, selecciona Siguiente.

> **Note**
>
> AWS Management Console Solo muestra los secretos de Secrets Manager con nombres que usan el `ecr-pullthroughcache/` prefijo. El secreto también debe estar en la misma cuenta y región en las que se creó la regla de caché de extracción.

b. Para crear un secreto nuevo, selecciona Crear un AWS secreto, haz lo siguiente y, a continuación, selecciona Siguiente.

   i. En Nombre secreto, especifique un nombre descriptivo para el secreto. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode.

   ii. Como nombre de usuario de GitHub Container Registry, especifique su nombre de usuario de GitHub Container Registry.

   iii. Para el token de acceso a GitHub Container Registry, especifique su token de acceso a GitHub Container Registry. Para obtener más información sobre cómo crear un token de GitHub acceso, consulta Administrar tus tokens de acceso personales en la GitHub documentación.

7. En la página del Paso 3: Especificar un destino, para Prefijo del repositorio de Amazon ECR, especifique el prefijo del espacio de nombres del repositorio que se utilizará al almacenar en caché las imágenes extraídas del registro público de origen y, a continuación, seleccione Siguiente.

   De forma predeterminada, se rellena un espacio de nombres pero también se puede especificar un espacio de nombres personalizado.

8. En la página del Paso 4: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.

9. Repita el paso anterior para cada caché de extracción que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para Microsoft Azure Container Registry

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. Desde la barra de navegación, elija la región en la que desea configurar su registro privado.
3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).
4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).
5. En la página del Paso 1: Especificar un origen, haga lo siguiente.

   a. Para Registry, elija Microsoft Azure Container Registry

   b. En URL del registro de origen, especifique el nombre de Microsoft Azure Container Registry y, a continuación, elija Siguiente.

   > **Important**
   >
   > Solo necesita especificar el prefijo, ya que el `.azurecr.io` sufijo se rellena en su nombre.

6. En la página del Paso 2: Configurar la autenticación, en Credenciales principales, debe almacenar las credenciales de autenticación de Microsoft Azure Container Registry en un secreto de AWS Secrets Manager . Puede especificar un secreto existente o utilizar la consola Amazon ECR para crear uno nuevo.

   a. Para usar un secreto existente, elija Usar un AWS secreto existente. En Nombre secreto, usa el menú desplegable para seleccionar tu secreto existente y, a continuación, selecciona Siguiente.

   > **Note**
   >
   > AWS Management Console Solo muestra los secretos de Secrets Manager con nombres que usan el `ecr-pullthroughcache/` prefijo. El secreto también debe estar en la misma cuenta y región en las que se creó la regla de caché de extracción.

   b. Para crear un secreto nuevo, selecciona Crear un AWS secreto, haz lo siguiente y, a continuación, selecciona Siguiente.

i. En Nombre secreto, especifique un nombre descriptivo para el secreto. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode.

ii. Para el nombre de usuario de Microsoft Azure Container Registry, especifique su nombre de usuario de Microsoft Azure Container Registry.

iii. Para el nombre de usuario de Microsoft Azure Container Registry, especifique su nombre de usuario de Microsoft Azure Container Registry. Para obtener más información sobre la creación de un token de acceso a Microsoft Azure Container Registry, consulte Crear token: portal en la documentación de Microsoft Azure.

7. En la página del Paso 3: Especificar un destino, para Prefijo del repositorio de Amazon ECR, especifique el prefijo del espacio de nombres del repositorio que se utilizará al almacenar en caché las imágenes extraídas del registro público de origen y, a continuación, seleccione Siguiente.

   De forma predeterminada, se rellena un espacio de nombres pero también se puede especificar un espacio de nombres personalizado.

8. En la página del Paso 4: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.

9. Repita el paso anterior para cada caché de extracción que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para GitLab Container Registry

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.

2. Desde la barra de navegación, elija la región en la que desea configurar su registro privado.

3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).

4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).

5. En el paso 1: especifique una página de origen, en Registry, elija GitLab Container Registry y, a continuación.

6. En la página Paso 2: Configurar la autenticación, en el caso de las credenciales de Upstream, debe guardar en AWS Secrets Manager secreto las credenciales de autenticación de GitLab Container Registry. Puede especificar un secreto existente o utilizar la consola Amazon ECR para crear uno nuevo.

a. Para usar un secreto existente, selecciona Usar un AWS secreto existente. En Nombre secreto, usa el menú desplegable para seleccionar tu secreto existente y, a continuación, selecciona Siguiente. Para obtener más información sobre la creación de un secreto de Secrets Manager mediante la consola de Secrets Manager, consulteAlmacenar en secreto las credenciales del repositorio principal AWS Secrets Manager.

> **Note**
>
> AWS Management Console Solo muestra los secretos de Secrets Manager con nombres que usan el `ecr-pullthroughcache/` prefijo. El secreto también debe estar en la misma cuenta y región en las que se creó la regla de caché de extracción.

b. Para crear un secreto nuevo, selecciona Crear un AWS secreto, haz lo siguiente y, a continuación, selecciona Siguiente.

   i. En Nombre secreto, especifique un nombre descriptivo para el secreto. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode.

   ii. Como nombre de usuario de GitLab Container Registry, especifique su nombre de usuario de GitLab Container Registry.

   iii. Para el token de acceso a GitLab Container Registry, especifique su token de acceso a GitLab Container Registry. Para obtener más información sobre cómo crear un token de acceso a GitLab Container Registry, consulta los tokens de acceso personal, los tokens de acceso grupal o los tokens de acceso a proyectos en la GitLab documentación.

7. En la página del Paso 3: Especificar un destino, para Prefijo del repositorio de Amazon ECR, especifique el prefijo del espacio de nombres del repositorio que se utilizará al almacenar en caché las imágenes extraídas del registro público de origen y, a continuación, seleccione Siguiente.

   De forma predeterminada, se rellena un espacio de nombres pero también se puede especificar un espacio de nombres personalizado.

8. En la página del Paso 4: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.

9. Repita el paso anterior para cada caché de extracción que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para el registro privado de Amazon ECR en su cuenta AWS

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que desee configurar los ajustes de registro privado.
3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).
4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).
5. En la página Paso 1: Especificar origen, en Registry, elija Amazon ECR Private y This account. En Región, seleccione la región para el registro original de Amazon ECR y, a continuación, elija Siguiente.
6. En la página Paso 2: Especificar los espacios de nombres, para el espacio de nombres de caché, elija si desea crear repositorios de caché extraíbles con un prefijo específico o sin prefijo. Si selecciona un prefijo específico, debe especificar un nombre de prefijo para usarlo como parte del espacio de nombres para almacenar en caché las imágenes del registro original.
7. En el caso del espacio de nombres upstream, elija si desea utilizar un prefijo específico que exista en el registro upstream. Si no selecciona ningún prefijo, puede extraerlo de cualquier repositorio del registro principal. Especifique el prefijo del repositorio ascendente si se le solicita y, a continuación, seleccione Siguiente.

   > **Note**
   >
   > Para obtener más información sobre la personalización de los espacios de nombres ascendentes y de caché, consulte. Personalización de los prefijos del repositorio para que el ECR extraiga la memoria caché

8. En la página del Paso 3: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.
9. Repita estos pasos para cada caché extraíble que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para el registro privado de Amazon ECR desde otra cuenta AWS

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. Desde la barra de navegación, elija la región en la que desea configurar su registro privado.

3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).
4. En la página Pull through cache configuration(Configuración de caché de extracción), elija Add rule (Agregar regla).
5. En la página Paso 1: Especificar origen, en Registro, elija Amazon ECR Private and Cross Account. En Región, seleccione la región para el registro original de Amazon ECR. En Cuenta, especifique el ID de AWS cuenta para el registro de Amazon ECR original y, a continuación, seleccione Siguiente.
6. En la página Paso 2: Especificar los permisos, en el caso de la función de IAM, seleccione una función para utilizarla para el acceso a la memoria caché entre cuentas y, a continuación, seleccione Crear.

> **Note**
>
> Asegúrese de seleccionar el rol de IAM que usa los permisos creados en. Se requieren políticas de IAM para transferir la memoria caché de ECR a ECR entre cuentas

7. En la página Paso 3: Especificar los espacios de nombres, para el espacio de nombres de caché, elija si desea crear repositorios de caché extraíbles con un prefijo específico o sin prefijo. Si selecciona un prefijo específico, debe especificar un nombre de prefijo para usarlo como parte del espacio de nombres para almacenar en caché las imágenes del registro original.
8. En el caso del espacio de nombres upstream, elija si desea utilizar un prefijo específico que exista en el registro upstream. Si no selecciona ningún prefijo, puede extraerlo de cualquier repositorio del registro principal. Especifique el prefijo del repositorio ascendente si se le solicita y, a continuación, seleccione Siguiente.

> **Note**
>
> Para obtener más información sobre la personalización de los espacios de nombres ascendentes y de caché, consulte. Personalización de los prefijos del repositorio para que el ECR extraiga la memoria caché

9. En la página del Paso 4: Revisar y crear, revise la configuración de la regla de caché de extracción y, a continuación, seleccione Crear.
10. Repita estos pasos para cada caché extraíble que desee crear. Las reglas de caché de extracción se crean por separado para cada región.

Para crear una regla de caché de extracción (AWS CLI)

Utilice el AWS CLI comando create-pull-through-cache-rule para crear una regla de extracción de caché para un registro privado de Amazon ECR. En el caso de los registros anteriores que requieren autenticación con secretos, debe almacenar las credenciales en un secreto de Secrets Manager. Para crear un secreto desde la consola de Secrets Manager, consulte Almacenar en secreto las credenciales del repositorio principal AWS Secrets Manager.

Se proporcionan los siguientes ejemplos para cada registro ascendente compatible.

Para Amazon ECR Public

En el siguiente ejemplo se crea una regla de caché de extracción para el registro público de Amazon ECR. Especifica un prefijo de repositorio `ecr-public`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `ecr-public/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --upstream-registry-url public.ecr.aws \
     --region us-east-2
```

Para Kubernetes Container Registry

En el siguiente ejemplo se crea una regla de extracción de caché para el registro público de Kubernetes. Especifica un prefijo de repositorio `kubernetes`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `kubernetes/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix kubernetes \
     --upstream-registry-url registry.k8s.io \
     --region us-east-2
```

Para Quay

El siguiente ejemplo crea una regla de extracción de caché para el registro público de Quay. Especifica un prefijo de repositorio `quay`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `quay/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix quay \
     --upstream-registry-url quay.io \
     --region us-east-2
```

Para Docker Hub

El siguiente ejemplo crea una regla de caché de extracción para el registro de Docker Hub. Especifica un prefijo de repositorio `docker-hub`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `docker-hub/`*`upstream-repository-name`*. Debe especificar el nombre de recurso de Amazon (ARN) completo del secreto que contiene las credenciales de Docker Hub.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix docker-hub \
     --upstream-registry-url registry-1.docker.io \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Para Container Registry GitHub

En el siguiente ejemplo, se crea una regla de extracción de caché para el GitHub Container Registry. Especifica un prefijo de repositorio `github`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `github/`*`upstream-repository-name`*. Debe especificar el nombre de recurso de Amazon (ARN) completo del secreto que contiene sus credenciales de GitHub Container Registry.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix github \
     --upstream-registry-url ghcr.io \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Para Microsoft Azure Container Registry

El siguiente ejemplo crea una regla de caché de extracción para el Microsoft Azure Container Registry. Especifica un prefijo de repositorio `azure`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de

`azure/`*`upstream-repository-name`*. Debe especificar el nombre de recurso de Amazon (ARN) completo del secreto que contiene las credenciales de Microsoft Azure Container Registry.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix azure \
     --upstream-registry-url myregistry.azurecr.io \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Para GitLab Container Registry

En el siguiente ejemplo, se crea una regla de extracción de caché para el GitLab Container Registry. Especifica un prefijo de repositorio `gitlab`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `gitlab/`*`upstream-repository-name`*. Debe especificar el nombre de recurso de Amazon (ARN) completo del secreto que contiene sus credenciales de GitLab Container Registry.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix gitlab \
     --upstream-registry-url registry.gitlab.com \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Para el registro privado de Amazon ECR en su cuenta AWS

El siguiente ejemplo crea una regla de extracción de caché para el registro privado de Amazon ECR para Cross-Region dentro de la misma AWS cuenta. Especifica un prefijo de repositorio `ecr`, lo que da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `ecr/`*`upstream-repository-name`*.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr \
     --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
     --region us-east-2
```

Para el registro privado de Amazon ECR desde otra cuenta AWS

El siguiente ejemplo crea una regla de extracción de caché para el registro privado de Amazon ECR para Cross-Region dentro de la misma AWS cuenta. Especifica un prefijo de repositorio `ecr`, lo que

da como resultado que cada repositorio se cree mediante la regla de caché de extracción para tener el esquema de nomenclatura de `ecr/upstream-repository-name`. Debe especificar el nombre de recurso de Amazon (ARN) completo de la función de IAM con los permisos creados en. Creación de una regla de caché de extracción en Amazon ECR

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix ecr \
 --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
 --custom-role-arn arn:aws:iam::aws_account_id:role/example-role \
 --region us-east-2
```

Pasos a seguir a continuación

Una vez que haya creado las reglas de caché de extracción, continúe con los siguientes pasos:

- Cree una plantilla de creación de repositorios. Puede utilizar una plantilla de creación de repositorios para definir la configuración que se aplicará a los repositorios creados por Amazon ECR en su nombre durante una acción de caché de extracción. Para obtener más información, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.
- Valide sus reglas de caché de extracción. Al validar una regla de caché de extracción, Amazon ECR establece una conexión de red con el registro principal, comprueba que puede acceder al secreto de Secrets Manager que contiene las credenciales del registro ascendente y que la autenticación se ha realizado correctamente. Para obtener más información, consulte Validación de reglas de caché de extracción en Amazon ECR.
- Comience a usar sus reglas de caché de extracción. Para obtener más información, consulte Extracción de una imagen con una regla de caché de extracción en Amazon ECR.

Validación de reglas de caché de extracción en Amazon ECR

Después de crear una regla de caché de extracción para los registros principales que requieren autenticación, puede validar que la regla funciona correctamente. Al validar una regla de caché de extracción, Amazon ECR establece una conexión de red con el registro principal, comprueba que puede acceder al secreto de Secrets Manager que contiene las credenciales del registro ascendente y verifica que la autenticación se ha realizado correctamente.

Antes de empezar a trabajar con las reglas de caché de extracción, verifique que dispone de los permisos de IAM adecuados. Para obtener más información, consulte Permisos de IAM necesarios para sincronizar un registro principal con un registro privado de Amazon ECR.

Para validar una regla de caché de extracción (AWS Management Console)

Los siguientes pasos muestran cómo crear una regla de caché de extracción mediante la consola de Amazon ECR.

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región que contiene la regla de caché de extracción que desea validar.
3. En el panel de navegación, elija Private registry (Registro privado), Pull through cache (Caché de extracción).
4. En la página de Configuración de caché de extracción, seleccione la regla de caché de extracción que desee validar. A continuación, usa el menú desplegable Acciones y selecciona Ver detalles.
5. En la página de detalles de la regla de caché de extracción, use el menú desplegable Acciones y seleccione Verificar la autenticación. Amazon ECR mostrará un banner con el resultado.
6. Repita estos pasos para cada regla de caché de extracción que desee validar.

Para validar una regla de caché de extracción (AWS CLI)

El AWS CLI comando validate-pull-through-cache-rule se utiliza para validar una regla de extracción de caché para un registro privado de Amazon ECR. En el siguiente ejemplo, se utiliza `ecr-public` el prefijo del espacio de nombres. Sustituya ese valor por el valor de prefijo de la regla de caché de extracción que desee validar.

```
aws ecr validate-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --region us-east-2
```

En la respuesta, el `isValid` parámetro indica si la validación se ha realizado correctamente o no. Si es `true`, Amazon ECR pudo acceder al registro principal y la autenticación se realizó correctamente. Si`false`, se ha producido un problema y la validación ha fallado. El `failure` parámetro indica la causa.

Extracción de una imagen con una regla de caché de extracción en Amazon ECR

En los ejemplos siguientes se muestra la sintaxis del comando que se debe utilizar al extraer una imagen mediante una regla de caché de extracción. Si aparece un error al extraer una imagen ascendente mediante una regla de caché de extracción, consulte Solución de problemas con la caché de extracción en Amazon ECR para ver los errores más habituales y cómo resolverlos.

Antes de empezar a trabajar con las reglas de caché de extracción, verifique que dispone de los permisos de IAM adecuados. Para obtener más información, consulte Permisos de IAM necesarios para sincronizar un registro principal con un registro privado de Amazon ECR.

> **Note**
>
> En los ejemplos siguientes se utilizan los valores de espacio de nombres del repositorio de Amazon ECR predeterminados que utilizan. AWS Management Console Asegúrese de utilizar el URI del repositorio privado de Amazon ECR que ha configurado.

Para Amazon ECR Public

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/
image_name:tag
```

Kubernetes Container Registry

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/kubernetes/repository_name/
image_name:tag
```

Quay

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/quay/repository_name/
image_name:tag
```

Docker Hub

Para ver las imágenes oficiales de Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

> **Note**
>
> En el caso de las imágenes oficiales de Docker Hub, se debe `/library` incluir el prefijo. Para todos los demás repositorios de Docker Hub, debes omitir el prefijo. `/library`

Para el resto de imágenes de Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

GitHub Registro de contenedores

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/
image_name:tag
```

Para Microsoft Azure Container Registry

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/
image_name:tag
```

GitLab Registro de contenedores

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

Almacenar en secreto las credenciales del repositorio principal AWS Secrets Manager

Al crear una regla de caché de extracción para un repositorio ascendente que requiera autenticación, debe almacenar las credenciales en un secreto de Secrets Manager. El uso de un secreto de Secrets Manager puede tener un coste. Para obtener más información, consulte Precios de AWS Secrets Manager.

Los siguientes procedimientos explican cómo crear un secreto de Secrets Manager para cada repositorio upstream compatible. Si lo desea, puede utilizar el flujo de trabajo de creación de reglas de caché de extracción en la consola de Amazon ECR para crear el secreto en lugar de crearlo mediante la consola Secrets Manager. Para obtener más información, consulte Creación de una regla de caché de extracción en Amazon ECR.

Docker Hub

Para crear un secreto de Secrets Manager para tus credenciales de Docker Hub (AWS Management Console)

1. Abra la consola de Secrets Manager en https://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.
3. En la página Elegir tipo de secreto, haga lo siguiente:
   a. En Secret type (Tipo de secreto), elija Other type of secret (Otro tipo de secreto).
   b. En pares clave/valor, cree dos filas para sus credenciales de Docker Hub. Puede almacenar hasta 65536 bytes en el secreto.
      i. Para el primer key/value par, especifique `username` como clave y su nombre de usuario de Docker Hub como valor.
      ii. Para el segundo key/value par, `accessToken` especifíquelo como clave y su token de acceso a Docker Hub como valor. Para obtener más información sobre cómo crear un token de acceso de Docker Hub, consulta Crear y administrar los tokens de acceso en la documentación de Docker.
   c. En el caso de la clave de cifrado, mantenga el valor predeterminado de aws/secretsmanager AWS KMS key y, a continuación, seleccione Siguiente. No se aplica ningún cargo por el uso de esta clave. Para obtener más información, consulte Cifrado y descifrado secretos en Secrets Manager en la Guía del AWS Secrets Manager usuario.

   > **Important**
   >
   > Debe utilizar la clave de `aws/secretsmanager` cifrado predeterminada para cifrar su secreto. Amazon ECR no permite usar una clave administrada por el cliente (CMK) para esto.

4. En la página Configurar el secreto, haga lo siguiente:

a. Ingrese un Nombre de secreto descriptivo y una Descripción. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode con el prefijo `ecr-pullthroughcache/`.

> **Important**
>
> El Amazon ECR AWS Management Console solo muestra los secretos de Secrets Manager cuyos nombres utilizan el `ecr-pullthroughcache/` prefijo.

b. (Opcional) En la sección Tags (Etiquetas), agregue etiquetas a su secreto. Para conocer las estrategias de etiquetado, consulte los secretos de Tag Secrets Manager en la Guía del AWS Secrets Manager usuario. No almacene información confidencial en etiquetas porque no están cifradas.

c. (Opcional) En Resource permissions (Permisos de recursos), para agregar una política de recursos a su secreto, elija Edit permissions (Editar permisos). Para obtener más información, consulte Adición de una política de permisos a un secreto en la Guía del usuario de AWS Secrets Manager .

d. (Opcional) En Replicar secreto, para replicar su secreto en otra persona Región de AWS, elija Replicar secreto. Puede replicar el secreto ahora o volver y replicarlo más tarde. Para obtener más información, consulte Replicar un secreto a otras regiones en la Guía del usuario de AWS Secrets Manager .

e. Elija Siguiente.

5. (Opcional) En la página Configure rotation (Configurar rotación), puede activar la rotación automática. También puede mantener la rotación desactivada por ahora y activarla más tarde. Para obtener más información, consulte Rotar secretos de Secrets Manager en la Guía del usuario de AWS Secrets Manager . Elija Siguiente.

6. En la página Review (Revisar), revise los detalles del secreto y, a continuación, elija Store (Almacenar).

   Secrets Manager vuelve a la lista de secretos. Si el nuevo secreto no aparece, elija el botón Refresh (Actualizar).

GitHub Container Registry

Para crear un secreto de Secrets Manager para tus credenciales de GitHub Container Registry (AWS Management Console)

1. Abra la consola de Secrets Manager en https://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.
3. En la página Elegir tipo de secreto, haga lo siguiente:
   a. En Secret type (Tipo de secreto), elija Other type of secret (Otro tipo de secreto).
   b. En pares clave/valor, cree dos filas para sus GitHub credenciales. Puede almacenar hasta 65536 bytes en el secreto.
      i. Para el primer key/value par, especifique `username` como clave y su GitHub nombre de usuario como valor.
      ii. Para el segundo key/value par, especifique `accessToken` como clave y su token de GitHub acceso como valor. Para obtener más información sobre cómo crear un token de GitHub acceso, consulte Administrar sus tokens de acceso personales en la GitHub documentación.
   c. En el caso de la clave de cifrado, mantenga el valor predeterminado de aws/secretsmanager AWS KMS key y, a continuación, seleccione Siguiente. No se aplica ningún cargo por el uso de esta clave. Para obtener más información, consulte Cifrado y descifrado secretos en Secrets Manager en la Guía del AWS Secrets Manager usuario.

      > **Important**
      >
      > Debe utilizar la clave de `aws/secretsmanager` cifrado predeterminada para cifrar su secreto. Amazon ECR no permite usar una clave administrada por el cliente (CMK) para esto.

4. En la página Configure secret (Configurar el secreto), haga lo siguiente:
   a. Ingrese un Nombre de secreto descriptivo y una Descripción. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode con el prefijo `ecr-pullthroughcache/`.

> **Important**
>
> El Amazon ECR AWS Management Console solo muestra los secretos de Secrets Manager cuyos nombres utilizan el `ecr-pullthroughcache/` prefijo.

b. (Opcional) En la sección Tags (Etiquetas), agregue etiquetas a su secreto. Para conocer las estrategias de etiquetado, consulte los secretos de Tag Secrets Manager en la Guía del AWS Secrets Manager usuario. No almacene información confidencial en etiquetas porque no están cifradas.

c. (Opcional) En Resource permissions (Permisos de recursos), para agregar una política de recursos a su secreto, elija Edit permissions (Editar permisos). Para obtener más información, consulte Adición de una política de permisos a un secreto en la Guía del usuario de AWS Secrets Manager .

d. (Opcional) En Replicar secreto, para replicar su secreto en otra persona Región de AWS, elija Replicar secreto. Puede replicar el secreto ahora o volver y replicarlo más tarde. Para obtener más información, consulte Replicar un secreto a otras regiones en la Guía del usuario de AWS Secrets Manager .

e. Elija Siguiente.

5. (Opcional) En la página Configure rotation (Configurar rotación), puede activar la rotación automática. También puede mantener la rotación desactivada por ahora y activarla más tarde. Para obtener más información, consulte Rotar secretos de Secrets Manager en la Guía del usuario de AWS Secrets Manager . Elija Siguiente.

6. En la página Review (Revisar), revise los detalles del secreto y, a continuación, elija Store (Almacenar).

   Secrets Manager vuelve a la lista de secretos. Si el nuevo secreto no aparece, elija el botón Refresh (Actualizar).

Microsoft Azure Container Registry

Para crear un secreto de Secrets Manager para sus credenciales de Microsoft Azure Container Registry (AWS Management Console)

1. Abra la consola de Secrets Manager en https://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.

3. En la página Elegir tipo de secreto, haga lo siguiente:

   a. En Secret type (Tipo de secreto), elija Other type of secret (Otro tipo de secreto).

   b. En pares clave/valor, cree dos filas para sus credenciales de GitHub. Puede almacenar hasta 65536 bytes en el secreto.

      i. Para el primer key/value par, especifique `username` como clave y su nombre de usuario de Microsoft Azure Container Registry como valor.

      ii. Para el segundo key/value par, especifique `accessToken` como clave y su token de acceso a Microsoft Azure Container Registry como valor. Para obtener más información sobre la creación de un token de acceso a Microsoft Azure Container Registry, consulte Crear token: portal en la documentación de Microsoft Azure.

   c. En el caso de la clave de cifrado, mantenga el valor predeterminado de aws/secretsmanager AWS KMS key y, a continuación, seleccione Siguiente. No se aplica ningún cargo por el uso de esta clave. Para obtener más información, consulte Cifrado y descifrado secretos en Secrets Manager en la Guía del AWS Secrets Manager usuario.

   > **Important**
   >
   > Debe utilizar la clave de `aws/secretsmanager` cifrado predeterminada para cifrar su secreto. Amazon ECR no permite usar una clave administrada por el cliente (CMK) para esto.

4. En la página Configure secret (Configurar el secreto), haga lo siguiente:

   a. Ingrese un Nombre de secreto descriptivo y una Descripción. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode con el prefijo `ecr-pullthroughcache/`.

   > **Important**
   >
   > El Amazon ECR AWS Management Console solo muestra los secretos de Secrets Manager cuyos nombres utilizan el `ecr-pullthroughcache/` prefijo.

   b. (Opcional) En la sección Tags (Etiquetas), agregue etiquetas a su secreto. Para conocer las estrategias de etiquetado, consulte los secretos de Tag Secrets Manager en la Guía del AWS Secrets Manager usuario. No almacene información confidencial en etiquetas porque no están cifradas.

c. (Opcional) En Resource permissions (Permisos de recursos), para agregar una política de recursos a su secreto, elija Edit permissions (Editar permisos). Para obtener más información, consulte Adición de una política de permisos a un secreto en la Guía del usuario de AWS Secrets Manager .

d. (Opcional) En Replicar secreto, para replicar su secreto en otra persona Región de AWS, elija Replicar secreto. Puede replicar el secreto ahora o volver y replicarlo más tarde. Para obtener más información, consulte Replicar un secreto a otras regiones en la Guía del usuario de AWS Secrets Manager .

e. Elija Siguiente.

5. (Opcional) En la página Configure rotation (Configurar rotación), puede activar la rotación automática. También puede mantener la rotación desactivada por ahora y activarla más tarde. Para obtener más información, consulte Rotar secretos de Secrets Manager en la Guía del usuario de AWS Secrets Manager . Elija Siguiente.

6. En la página Review (Revisar), revise los detalles del secreto y, a continuación, elija Store (Almacenar).

   Secrets Manager vuelve a la lista de secretos. Si el nuevo secreto no aparece, elija el botón Refresh (Actualizar).

GitLab Container Registry

Para crear un secreto de Secrets Manager para tus credenciales de GitLab Container Registry (AWS Management Console)

1. Abra la consola de Secrets Manager en https://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.
3. En la página Elegir tipo de secreto, haga lo siguiente:

   a. En Secret type (Tipo de secreto), elija Other type of secret (Otro tipo de secreto).

   b. En pares clave/valor, cree dos filas para sus GitLab credenciales. Puede almacenar hasta 65536 bytes en el secreto.

      i. Para el primer key/value par, especifique `username` como clave y su nombre de usuario de GitLab Container Registry como valor.

      ii. Para el segundo key/value par, especifique `accessToken` como clave y su token de acceso a GitLab Container Registry como valor. Para obtener más información

sobre cómo crear un token de acceso a GitLab Container Registry, consulte Tokens de acceso personal, Tokens de acceso grupal o Tokens de acceso de Project en la GitLab documentación.

c. En el caso de la clave de cifrado, mantenga el valor predeterminado de aws/secretsmanager AWS KMS key y, a continuación, seleccione Siguiente. No se aplica ningún cargo por el uso de esta clave. Para obtener más información, consulte Cifrado y descifrado secretos en Secrets Manager en la Guía del AWS Secrets Manager usuario.

> **Important**
>
> Debe utilizar la clave de `aws/secretsmanager` cifrado predeterminada para cifrar su secreto. Amazon ECR no permite usar una clave administrada por el cliente (CMK) para esto.

4. En la página Configure secret (Configurar el secreto), haga lo siguiente:

a. Ingrese un Nombre de secreto descriptivo y una Descripción. Los nombres de los secretos deben contener de 1 a 512 caracteres Unicode con el prefijo `ecr-pullthroughcache/`.

> **Important**
>
> El Amazon ECR AWS Management Console solo muestra los secretos de Secrets Manager cuyos nombres utilizan el `ecr-pullthroughcache/` prefijo.

b. (Opcional) En la sección Tags (Etiquetas), agregue etiquetas a su secreto. Para conocer las estrategias de etiquetado, consulte los secretos de Tag Secrets Manager en la Guía del AWS Secrets Manager usuario. No almacene información confidencial en etiquetas porque no están cifradas.

c. (Opcional) En Resource permissions (Permisos de recursos), para agregar una política de recursos a su secreto, elija Edit permissions (Editar permisos). Para obtener más información, consulte Adición de una política de permisos a un secreto en la Guía del usuario de AWS Secrets Manager .

d. (Opcional) En Replicar secreto, para replicar su secreto en otra persona Región de AWS, elija Replicar secreto. Puede replicar el secreto ahora o volver y replicarlo más tarde. Para obtener más información, consulte Replicar un secreto a otras regiones en la Guía del usuario de AWS Secrets Manager .

e. Elija Siguiente.

5. (Opcional) En la página Configure rotation (Configurar rotación), puede activar la rotación automática. También puede mantener la rotación desactivada por ahora y activarla más tarde. Para obtener más información, consulte Rotar secretos de Secrets Manager en la Guía del usuario de AWS Secrets Manager . Elija Siguiente.
6. En la página Review (Revisar), revise los detalles del secreto y, a continuación, elija Store (Almacenar).

   Secrets Manager vuelve a la lista de secretos. Si el nuevo secreto no aparece, elija el botón Refresh (Actualizar).

Personalización de los prefijos del repositorio para que el ECR extraiga la memoria caché

Las reglas de extracción de caché admiten tanto el prefijo de repositorio ecr como el prefijo de repositorio ascendente. El prefijo del repositorio ecr es el prefijo del espacio de nombres del repositorio en el registro de caché de Amazon ECR que está asociado a la regla. Todos los repositorios que utilizan este prefijo se convierten en repositorios de extracción con caché habilitada para el registro principal definido en la regla. Por ejemplo, el prefijo de se aplica a todos los repositorios que comiencen por. `prod` `prod/` Para aplicar una plantilla a todos los repositorios de tu registro que no tengan asociada una regla de extracción de caché, utilízala `ROOT` como prefijo.

> **Important**
>
> Siempre se aplica una suposición `/` al final del prefijo. Si especifica `ecr-public` como prefijo, Amazon ECR lo tratará como `ecr-public/`.

El prefijo del repositorio ascendente coincide con el nombre del repositorio ascendente. De forma predeterminada, está configurado en`ROOT`, lo que permite que coincida con cualquier repositorio anterior. Puede configurar el prefijo del repositorio ascendente solo cuando el prefijo del repositorio de Amazon ECR no tenga un valor. `ROOT`

En la siguiente tabla se muestra el mapeo entre los nombres de los repositorios de caché y los nombres de los repositorios ascendentes en función de sus configuraciones de prefijo en las reglas de extracción de caché.

Espacio de nombres de caché	Espacio de nombres ascendente	Relación de mapeo (repositorio de caché → repositorio ascendente)

ecr-public	ROOT (predeterminado)	`ecr-public/my-app/image1` → `my-app/image1`
		
		`ecr-public/my-app/image2` → `my-app/image2`

RAÍZ	RAÍZ	`my-app/image1` → `my-app/image1`
equipo-a	equipo-a	`team-a/myapp/image1` → `team-a/myapp/image1`
mi aplicación	aplicación ascendente	`my-app/image1` → `upstream-app/image1`

Solución de problemas con la caché de extracción en Amazon ECR

Cuando se extrae una imagen ascendente mediante una regla de caché de extracción, los errores que aparecen con más frecuencia son los siguientes.

El repositorio no existe

Un error que indica que el repositorio no existe suele deberse bien a que el repositorio no existe en el registro privado de Amazon ECR o bien a que no se ha otorgado el permiso `ecr:CreateRepository` a la entidad principal de IAM que realiza la extracción de la imagen ascendente. Para resolver este error, se debe verificar que el URI del repositorio del comando de extracción es correcto, que se han otorgado los permisos de IAM necesarios a la entidad principal que realiza la extracción de la imagen ascendente o que el repositorio donde se va a insertar la imagen ascendente se ha creado en el registro privado de Amazon ECR antes de realizar el extracción de la imagen ascendente. Para obtener más información sobre los permisos de IAM

necesarios, consulte Permisos de IAM necesarios para sincronizar un registro principal con un registro privado de Amazon ECR.

A continuación se muestra un ejemplo de este error.

```
Error response from daemon: repository 111122223333.dkr.ecr.us-east-1.amazonaws.com/
ecr-public/amazonlinux/amazonlinux not found: name unknown: The repository with
 name 'ecr-public/amazonlinux/amazonlinux' does not exist in the registry with id
 '111122223333'
```

Imagen solicitada no encontrada

Un error que indica que no se puede encontrar la imagen suele deberse bien a que la imagen no existe en el registro ascendente o bien a que no se ha otorgado el permiso `ecr:BatchImportUpstreamImage` a la entidad principal de IAM que realiza la extracción de la imagen ascendente, aunque el repositorio ya se va a crear en el registro privado de Amazon ECR. Para resolver este error, se debe verificar que la imagen ascendente y el nombre de la etiqueta de la imagen son correctos y existen, así como que se han otorgado los permisos de IAM necesarios a la entidad principal de IAM que realiza la extracción de la imagen ascendente. Para obtener más información sobre los permisos de IAM necesarios, consulte Permisos de IAM necesarios para sincronizar un registro principal con un registro privado de Amazon ECR.

A continuación se muestra un ejemplo de este error.

```
Error response from daemon: manifest for 111122223333.dkr.ecr.us-
east-1.amazonaws.com/ecr-public/amazonlinux/amazonlinux:latest not found: manifest
 unknown: Requested image not found
```

403 Prohibido cuando se extrae de un repositorio de Docker Hub

Cuando extraiga datos de un repositorio de Docker Hub que esté etiquetado como imagen oficial de Docker, debe incluir `/library/` en la URI que utilices. Por ejemplo, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/docker-hub/`**`library/`***`image_name:tag`*. Si omite las imágenes oficiales `/library/` de Docker Hub, se mostrará un `403 Forbidden` error al intentar extraer la imagen mediante una regla de extracción de caché. Para obtener más información, consulte Extracción de una imagen con una regla de caché de extracción en Amazon ECR.

A continuación se muestra un ejemplo de este error.

```
Error response from daemon: failed to resolve reference "111122223333.dkr.ecr.us-
west-2.amazonaws.com/docker-hub/amazonlinux:2023": pulling from host
 111122223333.dkr.ecr.us-west-2.amazonaws.com failed with status code [manifests
 2023]: 403 Forbidden
```

Replicación de imágenes privadas en Amazon ECR

Puede configurar el registro privado de Amazon ECR para que admita la replicación de los repositorios. Amazon ECR admite la replicación entre regiones y entre cuentas. Para que se produzca la replicación entre cuentas, la cuenta de destino debe configurar una política de permisos de registro con objeto de permitir que se produzca la replicación desde el registro de origen. Para obtener más información, consulte Permisos de registro privado en Amazon ECR.

Temas

- Consideraciones sobre la replicación de imágenes privadas
- Ejemplos de replicación de imágenes privadas en Amazon ECR
- Configuración de la replicación de imágenes privadas en Amazon ECR

Consideraciones sobre la replicación de imágenes privadas

Al utilizar la replicación de imágenes privadas, se debe tener en cuenta lo siguiente.

- Solo se replica el contenido del repositorio insertado en un repositorio una vez configurada la replicación. El contenido preexistente en un repositorio no se replica. Una vez configurada la replicación para un repositorio, Amazon ECR mantiene sincronizados el destino y el origen.
- El nombre del repositorio seguirá siendo el mismo en todas las regiones y cuentas cuando se produzca la replicación. Amazon ECR no admite el cambio del nombre del repositorio durante la replicación.
- La primera vez que se configura el registro privado para la replicación, Amazon ECR crea un rol de IAM vinculado a servicios en su nombre. El rol de IAM vinculado a servicios otorga al servicio de replicación de Amazon ECR el permiso necesario para crear repositorios y replicar imágenes en el registro. Para obtener más información, consulte Uso de roles vinculados a servicios para Amazon ECR.
- Para que se produzca la replicación entre cuentas, el destino del registro privado debe conceder permiso con objeto de permitir que el registro de origen replique sus imágenes. Para ello, se establece una política de permisos de registro privado. Para obtener más información, consulte Permisos de registro privado en Amazon ECR.
- Si se cambia la política de permisos de un registro privado para quitar un permiso, se pueden completar las replicaciones en curso concedidas anteriormente.

- Para que se produzca la replicación entre regiones, tanto la cuenta de origen como la de destino deben estar habilitadas en la región antes de que se lleve a cabo cualquier acción de replicación dentro o hacia esa región. Para obtener más información, consulte Administración de las regiones de AWS en la Referencia general de Amazon Web Services.
- No se admite la replicación entre regiones entre AWS particiones. Por ejemplo, no se puede replicar un repositorio `us-west-2` en `cn-north-1`. Para obtener más información sobre AWS las particiones, consulte el formato ARN en la Referencia AWS general.
- La configuración de replicación de un registro privado puede contener hasta 25 destinos únicos en todas las reglas, con un máximo de 10 reglas. Cada regla puede contener hasta 100 filtros. Esto permite especificar reglas independientes para repositorios que contienen imágenes utilizadas para la producción y las pruebas, por ejemplo.
- La configuración de replicación admite el filtrado de los repositorios de un registro privado que se replican especificando un prefijo de repositorio. Para ver un ejemplo, consulta Ejemplo: Configurar la replicación entre regiones mediante un filtro de repositorio.
- Una acción de replicación solo se produce una vez por cada inserción de imagen. Por ejemplo, si ha configurado la replicación entre regiones desde `us-west-2` hasta `us-east-1` y de `us-east-1` hasta `us-east-2`, una imagen insertada en `us-west-2` se replicará solo en `us-east-1`, y no se replicará de nuevo en `us-east-2`. Este comportamiento se aplica a la replicación entre regiones y entre cuentas.
- La mayoría de las imágenes se replican en menos de 30 minutos, pero en casos excepcionales la replicación puede tardar más.
- La replicación del registro no realiza ninguna acción de eliminación. Las imágenes y repositorios replicados se pueden eliminar manualmente cuando ya no los utilice.
- Las políticas de repositorio, incluidas las políticas de IAM, así como las políticas de ciclo de vida, no se replican ni tienen ningún efecto en otro repositorio que no sea para el que están definidas.
- La configuración del repositorio no se replica de forma predeterminada; puede replicarla mediante plantillas de creación de repositorios. Estos ajustes incluyen la mutabilidad de las etiquetas, el cifrado, los permisos de los repositorios y las políticas de ciclo de vida. Para obtener más información sobre las plantillas de creación de repositorios, consultePlantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.
- Si la inmutabilidad de etiquetas está habilitada en un repositorio y se replica una imagen que utiliza la misma etiqueta que una existente, la imagen se replica pero no incluirá la etiqueta duplicada. Esto podría provocar que la imagen se quedara sin etiquetar.

Ejemplos de replicación de imágenes privadas en Amazon ECR

Los siguientes ejemplos muestran casos de uso comunes para la replicación de imágenes privadas. Si configura la replicación mediante el AWS CLI, puede utilizar los ejemplos de JSON como punto de partida al crear el archivo JSON. Si configuras la replicación mediante el AWS Management Console, verás un JSON similar cuando revises tu regla de replicación en la página Revisar y enviar.

Ejemplo: Configurar la replicación entre regiones en una única región de destino

A continuación se muestra un ejemplo para configurar la replicación entre regiones en un único registro. En este ejemplo, se presupone que el ID de cuenta es `111122223333` y que está especificando esta configuración de replicación en una región distinta de `us-west-2`.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Ejemplo: Configurar la replicación entre regiones mediante un filtro de repositorio

A continuación se muestra un ejemplo con objeto de configurar la replicación entre regiones para repositorios que coincidan con un valor de nombre de prefijo. En este ejemplo, se presupone que el ID de cuenta es `111122223333` y que está especificando esta configuración de replicación en una región distinta de `us-west-1` y que tiene repositorios con un prefijo `prod`.

```
{
 "rules": [{
  "destinations": [{
```

```
    "region": "us-west-1",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "prod",
    "filterType": "PREFIX_MATCH"
   }]
  }]
}
```

Ejemplo: Configurar la replicación entre regiones en varias regiones de destino

A continuación se muestra un ejemplo para configurar la replicación entre regiones en un único registro. En este ejemplo se presupone que el ID de cuenta es `111122223333` y que está especificando esta configuración de replicación en una región distinta de `us-west-1` o `us-west-2`.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-1",
                    "registryId": "111122223333"
                },
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Ejemplo: Configurar la replicación entre cuentas

A continuación se muestra un ejemplo de configuración de la replicación entre cuentas para el registro. En este ejemplo se configura la replicación en la cuenta `444455556666` y en la región `us-west-2`.

> **Important**
>
> Para que se produzca la replicación entre cuentas, la cuenta de destino debe configurar una política de permisos de registro con objeto de permitir que se produzca la replicación. Para obtener más información, consulte Permisos de registro privado en Amazon ECR.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "444455556666"
                }
            ]
        }
    ]
}
```

Ejemplo: Especificar varias reglas en una configuración

A continuación se muestra un ejemplo de configuración de varias reglas de replicación para el registro. En este ejemplo se configura la replicación para la cuenta *111122223333*, con una regla que replica repositorios con un prefijo `prod` en la región `us-west-2` y repositorios con un prefijo `test` en la región `us-east-2`. Una configuración de replicación puede incluir hasta 10 reglas, y cada regla puede especificar hasta 25 destinos.

```
{
 "rules": [{
   "destinations": [{
    "region": "us-west-2",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "prod",
    "filterType": "PREFIX_MATCH"
   }]
  },
  {
   "destinations": [{
```

```
    "region": "us-east-2",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "test",
    "filterType": "PREFIX_MATCH"
   }]
  }
 ]
}
```

Configuración de la replicación de imágenes privadas en Amazon ECR

Configure la replicación por región para su registro privado. Puede configurar la replicación entre regiones o entre cuentas.

Para ver ejemplos de cómo se usa habitualmente la replicación, consulte Ejemplos de replicación de imágenes privadas en Amazon ECR.

Configuración de las opciones de replicación del registro (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, elija la región en la que se va a configurar la política de replicación del registro.
3. En el panel de navegación, elija Private registry (Registro privado).
4. En la página de Registro privado, elija Configuración y, a continuación, elija Editar en Configuración de replicación.
5. En la página Replication (Replicación), elija Add replication rule (Agregar regla de replicación).
6. En la página Destination types (Tipos de destino), elija si desea habilitar la replicación entre regiones, entre cuentas o ambas y, a continuación, elija Next (Siguiente).
7. Si está habilitada la replicación entre regiones, en Configure destination regions (Configurar regiones de destino) elija una o varias Destination regions (Regiones de destino) y luego Next (Siguiente).
8. Si está habilitada la replicación entre cuentas, en Cross-account replication (Replicación entre cuentas) elija la configuración de replicación entre cuentas del registro. En Destination account (Cuenta de destino), ingrese el ID de cuenta de la cuenta de destino y una o varias Destination

regions (Regiones de destino) en las que se vaya a replicar. Elija Destination account + (Cuenta de destino +) para configurar cuentas adicionales como destinos de replicación.

> **Important**
>
> Para que se produzca la replicación entre cuentas, la cuenta de destino debe configurar una política de permisos de registro con objeto de permitir que se produzca la replicación. Para obtener más información, consulte Permisos de registro privado en Amazon ECR.

9. (Opcional) En la página Add filters (Agregar filtros), especifique uno o más filtros para la regla de replicación y, a continuación, elija Add (Agregar). Repita este paso para cada filtro que desee asociar a la acción de replicación. Se debe especificar un filtro como prefijo del nombre del repositorio. Si no se agrega ningún filtro, se replicará el contenido de todos los repositorios. Elija Next (Siguiente) una vez que se hayan agregado todos los filtros.
10. En la página Review and submit (Revisar y enviar), revise la configuración de regla de replicación y luego elija Submit rule (Enviar regla).

Configuración de las opciones de replicación del registro (AWS CLI)

1. Cree un archivo JSON que incluya las reglas de replicación que se definirán para el registro. Una configuración de replicación puede contener hasta 10 reglas, con hasta 25 destinos exclusivos para todas las reglas y 100 filtros por cada regla. Para configurar la replicación entre regiones en de su propia cuenta, especifique su propio ID de cuenta. Para obtener más ejemplos, consulte Ejemplos de replicación de imágenes privadas en Amazon ECR.

```
{
 "rules": [{
  "destinations": [{
   "region": "destination_region",
   "registryId": "destination_accountId"
  }],
  "repositoryFilters": [{
   "filter": "repository_prefix_name",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

2. Cree una configuración de replicación para el registro.

```
aws ecr put-replication-configuration \
     --replication-configuration file://replication-settings.json \
     --region us-west-2
```

3. Confirme la configuración del registro.

```
aws ecr describe-registry \
     --region us-west-2
```

Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación

Utilice plantillas de creación de repositorios de Amazon ECR para definir la configuración que se aplicará a los repositorios creados por Amazon ECR en su nombre. La configuración de una plantilla de creación de repositorios solo se aplica durante la creación del repositorio y no tiene ningún efecto en los repositorios existentes ni en los repositorios creados mediante cualquier otro método. Actualmente, las plantillas de creación de repositorios se pueden utilizar para aplicar la configuración durante la creación del repositorio para las siguientes características:

- Caché de extracción
- Replicación

Las plantillas de creación de repositorio no se admiten en las siguientes regiones:

- China (Pekín) (`cn-north-1`)
- China (Ningxia) (`cn-northwest-1`)
- AWS GovCloud (EE. UU.-Este) (`us-gov-east-1`)
- AWS GovCloud (EEUU-Oeste) () `us-gov-west-1`

Cómo funcionan las plantillas de creación de repositorios

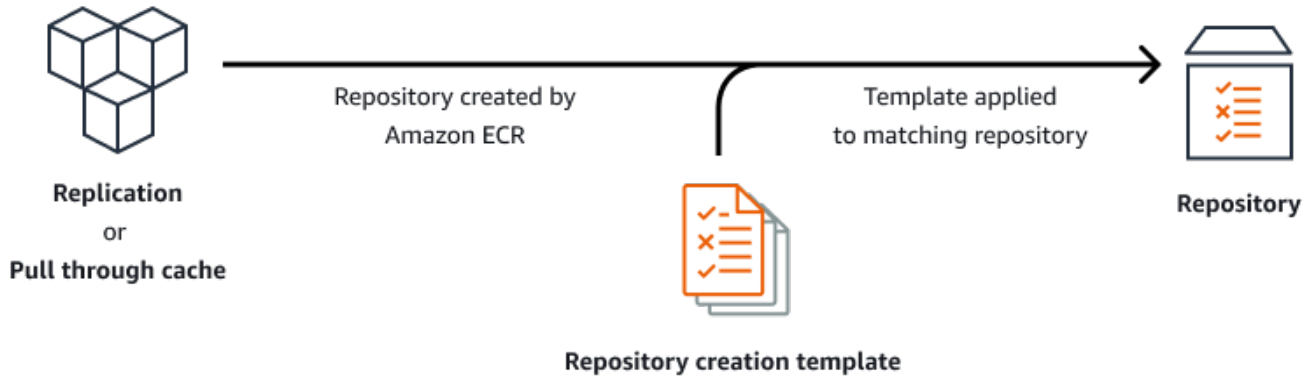
Hay ocasiones en las que Amazon ECR necesita crear un nuevo repositorio privado en su nombre. Por ejemplo:

- La primera vez que utilice una regla de caché de extracción para recuperar el contenido de un repositorio principal y almacenarlo en su registro privado de Amazon ECR.
- Cuando desee que Amazon ECR replique un repositorio en otra región o cuenta.

Cuando no hay una plantilla de creación de repositorios que se ajuste a la regla de caché de extracción o repositorio replicado, Amazon ECR utiliza la configuración predeterminada para el nuevo repositorio. Estas configuraciones predeterminadas incluyen desactivar la inmutabilidad de las etiquetas, utilizar el cifrado `AES-256` y no aplicar ninguna política de repositorio o ciclo de vida.

El uso de una plantilla de creación de repositorios le permite definir los ajustes que Amazon ECR aplica a los nuevos repositorios creados mediante las acciones de caché de extracción y replicación. Puede definir la inmutabilidad de las etiquetas, la configuración del cifrado, los permisos de los repositorios, la política de ciclo de vida y las etiquetas de recursos para los nuevos repositorios.

En el siguiente diagrama, se muestra el flujo de trabajo que Amazon ECR usa cuando se usa una plantilla de creación de repositorio con una acción de caché de extracción.



A continuación se describe en detalle cada parámetro de una plantilla de creación de repositorios.

Prefijo

El prefijo es el prefijo del espacio de nombres del repositorio que se va a asociar a la plantilla. A todos los repositorios creados con este prefijo se les aplicará la configuración definida en esta plantilla. Por ejemplo, el prefijo de `prod` se aplicaría a todos los repositorios que comiencen por `prod/`. De forma similar, el prefijo de `prod/team` se aplicaría a todos los repositorios que comiencen por `prod/team/`. En un registro que contiene dos plantillas, si una tiene el prefijo «prod» y la otra el prefijo "/». prod/team", the template with the prefix "prod/team" will be applied to all repositories whose names start with "prod/team

Para aplicar una plantilla a todos los repositorios de su registro que no tengan una plantilla de creación asociada, puede utilizar `ROOT` como prefijo.

> **Important**
>
> Siempre se aplica una suposición `/` al final del prefijo. Si especifica `ecr-public` como prefijo, Amazon ECR lo tratará como `ecr-public/`. Cuando utilice una regla de caché de extracción, el prefijo de repositorio que especifique durante la creación de la regla es el que debe especificar también como prefijo de la plantilla de creación de repositorios.

Descripción

Esta descripción de la plantilla es opcional y se utiliza para describir el propósito de la plantilla de creación de repositorios.

Aplicar a

La configuración aplicada determina qué repositorios creados por Amazon ECR se crearán con esta plantilla. Los valores válidos son `PULL_THROUGH_CACHE` y `REPLICATION`. Por ejemplo, la primera vez que utilice una regla de caché de extracción para recuperar el contenido de un repositorio de origen y almacenarlo en su registro privado de Amazon ECR. Cuando no hay una plantilla de creación de repositorios que se ajuste a la regla de caché de extracción, Amazon ECR utiliza la configuración predeterminada para el nuevo repositorio.

Rol de creación de repositorios

El rol de creación de repositorios es un ARN del rol de IAM que Amazon ECR asumirá al crear y configurar repositorios mediante plantillas de creación de repositorios. Esta función debe proporcionarse cuando se utilizan etiquetas de repositorio and/or KMS en la plantilla; de lo contrario, la creación del repositorio fallará.

Mutabilidad de etiquetas de imágenes

La configuración de mutabilidad de etiquetas que se utilizará en los repositorios creados con la plantilla. Si se omite este parámetro, se utilizará la configuración predeterminada de MUTABLE, lo que permitirá sobrescribir las etiquetas de imágenes. Esta es la configuración recomendada para las plantillas utilizadas en los repositorios creados mediante acciones de caché de extracción. Esto garantiza que Amazon ECR pueda actualizar las imágenes almacenadas en caché cuando las etiquetas sean las mismas.

Si se especifica INMUTABLE, todas las etiquetas de imagen dentro del repositorio serán inmutables, lo que evitará que se sobrescriban.

La configuración de cifrado

> **Important**
>
> El cifrado de doble capa del lado del servidor con AWS KMS (DSSE-KMS) solo está disponible en las regiones. AWS GovCloud (US)

La configuración de cifrado que se utilizará en los repositorios creados con la plantilla.

Si utiliza el tipo de cifrado de KMS, el contenido del repositorio se cifrará mediante el cifrado de lado del servidor con la clave de AWS Key Management Service almacenada en AWS KMS. Cuando utiliza AWS KMS para cifrar sus datos, puede utilizar la AWS KMS clave AWS gestionada predeterminada para Amazon ECR o especificar su propia AWS KMS clave, que ya ha creado. También puede optar por utilizar el cifrado de una o dos capas con. AWS KMS Para obtener más información, consulte Cifrado en reposo . Si utiliza el tipo de cifrado KMS junto con la replicación entre regiones, es posible que necesite permisos adicionales. Para obtener más información, consulte Crear una política de claves de KMS para la replicación.

Si utiliza el tipo de encriptación AES256, Amazon ECR utiliza el cifrado del lado del servidor con claves de cifrado administradas por Amazon S3 que cifran las imágenes del repositorio mediante un algoritmo de cifrado AES-256. Para obtener más información, consulte Protección de datos mediante el cifrado del lado del servidor con las claves de cifrado administradas por Amazon S3 (SSE-S3) en la Guía del usuario de Amazon Simple Storage Service.

Permisos de repositorio

La política de repositorios que se aplicará a los repositorios creados con la plantilla. Una política de repositorio utiliza permisos basados en recursos para controlar el acceso a un repositorio. Los permisos basados en recursos permiten especificar qué usuarios o roles de IAM tienen acceso a un repositorio y qué acciones pueden realizar en él. De forma predeterminada, solo la AWS cuenta que creó el repositorio tiene acceso a un repositorio. Puede aplicar un documento de política para otorgar o denegar permisos adicionales a su repositorio. Para obtener más información, consulte Políticas de repositorios privados en Amazon ECR.

Política del ciclo de vida de repositorio

La política de ciclo de vida que se utilizará para los repositorios creados con la plantilla. Una política de ciclo de vida proporciona más control sobre la administración del ciclo de vida de las imágenes en un repositorio privado. Una política de ciclo de vida incluye una o varias reglas, en la que cada regla define una acción de Amazon ECR. Esto proporciona una forma de automatizar la limpieza de sus imágenes de contenedor, cuyo vencimiento se produce en función de su antigüedad o recuento. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.

Etiquetas de recursos

Las etiquetas de recursos son metadatos que se aplican a los repositorios para ayudarlo a categorizarlos y organizarlos. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Si utiliza plantillas de creación de repositorios con replicación entre regiones, este permiso se debe aplicar a la política de registro de destino.

Crear una plantilla de creación de repositorios en Amazon ECR

Puede crear una plantilla de creación de repositorio para definir la configuración que se utilizará en los repositorios creados por Amazon ECR en su nombre durante las acciones de caché de extracción o de replicación. Una vez creada la plantilla de creación de repositorios, se aplicará la configuración a todos los repositorios nuevos que se creen. Esto no tiene ningún efecto en ningún repositorio creado anteriormente.

Al configurar un repositorio con plantillas, puede especificar las claves KMS y las etiquetas de recursos. Si pretende utilizar claves de KMS, etiquetas de recursos o una combinación de ambas en una o más plantillas, tiene que realizar lo siguiente:

- Cree una política personalizada para las plantillas de creación de repositorios.
- Cree un rol de IAM para las plantillas de creación de repositorios.

Una vez finalizada la configuración, puede adjuntar el rol personalizado a plantillas específicas de su registro.

Permisos de IAM para crear plantillas de creación de repositorio

Los siguientes permisos son necesarios para que una entidad principal de IAM administre las plantillas de creación de repositorios. Este permiso debe otorgarse a través de una política de IAM basada en identidad.

- `ecr:CreateRepositoryCreationTemplate`: otorga permiso para crear una plantilla de creación de repositorio.
- `ecr:UpdateRepositoryCreationTemplate`: otorga permiso para actualizar una plantilla de creación de repositorio.
- `ecr:DescribeRepositoryCreationTemplates`: otorga permiso para enumerar plantillas de creación de repositorio en un registro.
- `ecr:DeleteRepositoryCreationTemplate`: otorga permiso para eliminar una plantilla de creación de repositorio.
- `ecr:CreateRepository`: otorga permiso para crear un repositorio de Amazon ECR.
- `ecr:PutLifecyclePolicy`: otorga permiso para crear una política de ciclo de vida y aplicarla a un repositorio. Este permiso solo es necesario si la plantilla de creación del repositorio incluye una política de ciclo de vida.

- `ecr:SetRepositoryPolicy`: otorga permiso para crear una política de permisos para un repositorio. Este permiso solo es necesario si la plantilla de creación del repositorio incluye una política de repositorio.
- `iam:PassRole`: otorga permiso para permitir que una entidad transfiera un rol a un servicio o a una aplicación. Este permiso es necesario para los servicios y las aplicaciones que necesitan adoptar un rol con el fin de realizar acciones en su nombre.

Cree una política personalizada para las plantillas de creación de repositorios

Puede utilizarla AWS Management Console para definir una política que se asociará posteriormente a un rol de IAM. Este rol de IAM se puede utilizar luego como un rol de creación de repositorios al configurar una plantilla de creación de repositorios.

AWS Management Console

Para utilizar el editor de políticas de JSON y crear una política personalizada para las plantillas de creación de repositorios.

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en. https://console.aws.amazon.com/iam/
2. En el panel de navegación de la izquierda, elija Políticas.
3. Elija Crear política.
4. En la sección Editor de políticas, seleccione la opción JSON.
5. Introduzca la siguiente política en el campo de JSON.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "ecr:CreateRepository",
                    "ecr:ReplicateImage",
                    "ecr:TagResource"
                ],
                "Resource": "*"
            },
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "kms:CreateGrant",
                    "kms:RetireGrant",
                    "kms:DescribeKey"
                ],
                "Resource": "*"
            }
        ]
}
```

6. Resuelva las advertencias de seguridad, errores o advertencias generales que se generen durante la validación de la política y luego elija Siguiente.
7. Cuando haya terminado de agregar permisos a la política, seleccione Siguiente.
8. En la página Revisar y crear, escriba el Nombre de la política y la Descripción (opcional) para la política que está creando. Revise los Permisos definidos en esta política para ver los permisos que concede la política.
9. Elija Crear política para guardar la nueva política.
10. Cree un rol para asignar esta política a la plantilla de creación, consulte Cree un rol de IAM para las plantillas de creación de repositorios.

Cree un rol de IAM para las plantillas de creación de repositorios

Puede utilizarla AWS Management Console para crear una función que Amazon ECR pueda utilizar cuando especifique la función de creación de repositorios en una plantilla de creación de repositorios que utilice etiquetas de repositorio o KMS en una plantilla.

AWS Management Console

Para crear un rol.

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en. https://console.aws.amazon.com/iam/
2. En el panel de navegación de la consola, elija Roles y, a continuación, seleccione Crear rol.
3. Elija el tipo de rol Política de confianza personalizada.
4. En la sección Política de confianza personalizada, pegue la política de confianza personalizada que se indica a continuación:

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "ecr.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
    ]
}
```

5. Elija Siguiente.
6. En la página Añadir permisos, seleccione la casilla de verificación situada junto a la política personalizada que creó anteriormente en la lista de Políticas de permisos y seleccione Siguiente.
7. Escriba un nombre para el rol en Nombre de rol. Los nombres de los roles deben ser únicos dentro de su. Cuenta de AWS Cuando se utiliza un nombre de rol en una política o como parte de un ARN, el nombre del rol distingue entre mayúsculas y minúsculas. Cuando los clientes ven un nombre de rol en la consola, por ejemplo, durante el proceso de inicio de sesión, el nombre del rol no distingue entre mayúsculas y minúsculas. Dado que varias entidades pueden hacer referencia al rol, no se puede editar el nombre del rol una vez que se crea.
8. (Opcional) En Descripción, ingrese una descripción para el nuevo rol.
9. Revise el rol y, a continuación, seleccione Crear rol.

Cree una plantilla de creación de repositorios

Una vez que haya completado los requisitos previos necesarios para las plantillas, puede proceder a crear las plantillas de creación de repositorios.

AWS Management Console

Crear una plantilla de creación de repositorios (AWS Management Console)

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.

2. En la barra de navegación, elija la región en la que crear la plantilla de creación del repositorio.
3. En el panel de navegación, elija Registro privado, Plantillas de creación de repositorios.
4. En la página de plantillas de creación de repositorios, seleccione Crear plantilla.
5. En la página del Paso 1: Definir una plantilla, en Detalles de la plantilla, elija Un prefijo específico para aplicar la plantilla a un prefijo de espacio de nombres de repositorio específico o elija Cualquier prefijo de su registro de ECR para aplicar la plantilla a todos los repositorios que no coincidan con ninguna otra plantilla de la región.

   a. Si elige Un prefijo específico, especifique el prefijo del espacio de nombres del repositorio al que se va a aplicar la plantilla en Prefijo. Siempre se aplica una suposición `/` al final del prefijo. Por ejemplo, el prefijo de `prod` se aplicaría a todos los repositorios que comiencen por `prod/`. De forma similar, el prefijo de `prod/team` se aplicaría a todos los repositorios que comiencen por `prod/team/`.

   b. Si selecciona Cualquier prefijo en su registro ECR, el Prefijo se establecerá en `ROOT`.
6. En aplicar a, especifique a qué flujos de trabajo de Amazon ECR se aplicará esta plantilla. Las opciones son `PULL_THROUGH_CACHE` y `REPLICATION`.
7. En Descripción de la plantilla, especifique una descripción opcional para la plantilla y, a continuación, seleccione Siguiente.
8. En la página del Paso 2: Agregar la configuración de creación de repositorios, especifique la configuración del repositorio que se aplicará a los repositorios creados con la plantilla.

   a. Para Mutabilidad de la etiqueta de imagen, elija la configuración de mutabilidad de la etiqueta que desea utilizar. Para obtener más información, consulte Cómo impedir que se sobrescriban las etiquetas de imagen en Amazon ECR.

      - Mutable: elija esta opción si desea que se sobrescriban las etiquetas de las imágenes. Se recomienda para los repositorios que utilizan acciones de extracción de caché para garantizar que Amazon ECR pueda actualizar las imágenes almacenadas en caché. Además, para deshabilitar las actualizaciones de etiquetas para algunas etiquetas mutables, introduzca los nombres de las etiquetas o utilice caracteres comodín (*) para hacer coincidir varias etiquetas similares en el cuadro de texto de exclusión de etiquetas mutables.
      - Inmutable: elige esta opción si quieres evitar que las etiquetas de las imágenes se sobrescriban y se aplica a todas las etiquetas y exclusiones del repositorio al insertar una imagen con una etiqueta existente. Amazon ECR devuelve un

`ImageTagAlreadyExistsException` si intentas insertar una imagen con una etiqueta existente. Además, para habilitar las actualizaciones de etiquetas para algunas etiquetas inmutables, introduzca los nombres de las etiquetas o utilice caracteres comodín (*) para hacer coincidir varias etiquetas similares en el cuadro de texto de exclusión de etiquetas inmutables.

b. Para Configuración del cifrado, elija el ajuste de cifrado que desee utilizar. Para obtener más información, consulte Cifrado en reposo.

   Cuando se selecciona AES-256, Amazon ECR utiliza cifrado del lado del servidor con claves de cifrado administradas por Amazon Simple Storage Service que cifra sus datos en reposo mediante un algoritmo de cifrado AES-256 estándar de la industria. Esto se ofrece sin costo adicional.

   Cuando se selecciona AWS KMS, Amazon ECR utiliza el cifrado del lado del servidor con las claves almacenadas en AWS Key Management Service (AWS KMS). Cuando utiliza AWS KMS para cifrar sus datos, puede utilizar la clave AWS gestionada predeterminada, que gestiona Amazon ECR, o especificar su propia AWS KMS clave, que se denomina clave gestionada por el cliente.

   > **Note**
   >
   > La configuración de cifrado de un repositorio no se puede cambiar una vez creado el repositorio.

c. En Permisos del repositorio, especifique la política de permisos del repositorio que se aplicará a los repositorios creados con esta plantilla. Si lo desea, puede utilizar el menú desplegable para seleccionar uno de los ejemplos de JSON para los casos de uso más comunes. Para obtener más información, consulte Políticas de repositorios privados en Amazon ECR.

d. En Política de ciclo de vida del repositorio, especifique la política de ciclo de vida del repositorio que se aplicará a los repositorios creados con esta plantilla. Si lo desea, puede utilizar el menú desplegable para seleccionar uno de los ejemplos de JSON para los casos de uso más comunes. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.

e. En el caso de las AWS etiquetas de repositorio, especifique los metadatos, en forma de pares clave-valor, que desee asociar a los repositorios creados con esta plantilla y, a

continuación, seleccione Siguiente. Para obtener más información, consulte Etiquetado de un repositorio privado en Amazon ECR.

f. Para el rol de creación de repositorios, seleccione un rol de IAM personalizado desde el menú desplegable con el fin de utilizarlo en las plantillas de creación de repositorios cuando se usen etiquetas de repositorio o KMS en la plantilla (para obtener más información, consulte Cree un rol de IAM para las plantillas de creación de repositorios). A continuación, presione Siguiente.

9. En la página del Paso 3: Revisar y crear, revise la configuración que especificó para la plantilla de creación del repositorio. Seleccione la opción Editar para realizar cambios. Cuando haya terminado, elija Crear.

AWS CLI

El create-repository-creation-template AWS CLI comando se utiliza para crear una plantilla de creación de repositorios para su registro privado.

Crear una plantilla de creación de repositorios (AWS CLI)

1. Utilice el AWS CLI para generar un esqueleto para el create-repository-creation-templatecomando.

```
aws ecr create-repository-creation-template \
    --generate-cli-skeleton
```

El resultado del comando mostrará la sintaxis completa de la plantilla de creación del repositorio.

```
{
"appliedFor":[""], // string array, but valid are PULL_THROUGH_CACHE and
 REPLICATION
"prefix": "string",
    "description": "string",
    "imageTagMutability":
 "MUTABLE"|"IMMUTABLE"|"IMMUTABLE_WITH_EXCLUSION"|"MUTABLE_WITH_EXCLUSION",
    "imageTagMutabilityExclusionFilters": [
        "filterType": "WILDCARD",
        "filter": "string"
    ],
    "repositoryPolicy": "string",
```

```
    "lifecyclePolicy": "string"
"encryptionConfiguration": {
"encryptionType": "AES256"|"KMS",
        "kmsKey": "string"
    },
    "resourceTags": [
        {
"Key": "string",
            "Value": "string"
        }
    ],
    "customRoleArn": "string", // must be a valid IAM Role ARN
}
```

2. Cree un archivo denominado `repository-creation-template.json` con el resultado del paso anterior. Esta plantilla establece una clave de cifrado de KMS para cualquier repositorio creado `prod/*` con una política de repositorio que permita enviar y extraer imágenes a repositorios futuros, establece una política de ciclo de vida que caducará las imágenes con más de dos semanas de antigüedad y establece una función personalizada que permitirá a ECR acceder a la clave de KMS y asignar la etiqueta de recurso a los repositorios `examplekey` futuros.

```
{
"prefix": "prod",
    "description": "For repositories cached from my PTC rule and in my
 replication configuration that start with 'prod/'",
    "appliedFor": ["PULL_THROUGH_CACHE","REPLICATION"],
    "encryptionConfiguration": {
"encryptionType": "KMS",
        "kmsKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-
cdef-example11111"
    },
    "resourceTags": [
        {
"Key": "examplekey",
            "Value": "examplevalue"
        }
    ],
    "imageTagMutability": "IMMUTABLE_WITH_EXCLUSION",
    "imageTagMutabilityExclusionFilters": [
      {
      "filterType": "WILDCARD",
```

```
      "filter": "latest"
      },
      {
      "filterType": "WILDCARD",
      "filter": "beta*"
      }
    ]
    "repositoryPolicy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid
\":\"AllowPushPullIAMRole\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":
\"arn:aws:iam::111122223333:user\/IAMusername\"},\"Action\":[\"ecr:BatchGetImage
\",\"ecr:BatchCheckLayerAvailability\",\"ecr:CompleteLayerUpload\",
\"ecr:GetDownloadUrlForLayer\",\"ecr:InitiateLayerUpload\",\"ecr:PutImage\",
\"ecr:UploadLayerPart\"]}]}",
    "lifecyclePolicy": "{\"rules\":[{\"rulePriority\":1,\"description\":\"Expire
 images older than 14 days\",\"selection\":{\"tagStatus\":\"any\",\"countType
\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},\"action\":
{\"type\":\"expire\"}}]}",
    "customRoleArn": "arn:aws:iam::111122223333:role/myRole"
}
```

3. Para crear una plantilla de creación de repositorios, utilice el siguiente comando. Asegúrese de especificar el nombre del archivo de configuración que creó en el paso anterior en lugar del `repository-creation-template.json` en el ejemplo siguiente.

```
aws ecr create-repository-creation-template \
    --cli-input-json file://repository-creation-template.json
```

Actualización de una plantilla de creación de repositorios

Puede editar una plantilla de creación de repositorios en el caso de que necesite cambiar sus configuraciones. Una vez que edite la plantilla de creación de repositorios, las nuevas configuraciones se aplicarán a la plantilla existente.

> **Important**
>
> Esto no tiene ningún efecto en ningún repositorio creado anteriormente.

AWS Management Console

Editar una plantilla de creación de repositorios (AWS Management Console)

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que se encuentra la plantilla de creación del repositorio que desea editar.
3. En el panel de navegación, elija Registro privado y, a continuación, Configuración.
4. En la barra de navegación, elija las Plantillas de creación de repositorios.
5. En la página de Plantillas de creación de repositorios, seleccione la plantilla de creación de repositorios que desea editar.
6. Desde el menú desplegable Acciones, elija Editar.
7. Revise y actualice los ajustes de configuración.
8. Elija actualizar para aplicar las nuevas configuraciones de plantillas de creación.

AWS CLI

Editar una plantilla de creación de repositorios (AWS CLI)

- Utilice el update-repository-creation-templatecomando para actualizar una plantilla de creación de repositorios existente. Debe especificar el valor `prefix` de la plantilla. En el siguiente ejemplo, se actualiza una plantilla de creación de repositorios con el prefijo `prod`.

```
aws ecr update-repository-creation-template \
     --prefix prod \
     --image-tag-mutability="IMMUTABLE_WITH_EXCLUSION" \
     --image-tag-mutability-exclusion-filters filterType=WILDCARD, filter=latest
```

  El resultado del comando mostrará los detalles de la plantilla de creación de repositorios actualizada.

Eliminar una plantilla de creación de repositorios en Amazon ECR

Puede eliminar una plantilla de creación de repositorios si ha terminado de utilizarla. Una vez eliminada una plantilla de creación de repositorios, los repositorios recién creados con el prefijo asociado durante una acción de extracción de caché o replicación heredarán la configuración

predeterminada, a menos que se encuentre otra plantilla coincidente. Para obtener más información, consulte Cómo funcionan las plantillas de creación de repositorios.

> **Important**
>
> Esto no tiene ningún efecto en ningún repositorio creado anteriormente.

AWS Management Console

Eliminar una plantilla de creación de repositorios (AWS Management Console)

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que se encuentra la plantilla de creación del repositorio que desea eliminar.
3. En el panel de navegación, elija Registro privado, Plantillas de creación de repositorios.
4. En la página de plantillas de creación de repositorios, seleccione la plantilla de creación de repositorios que desee eliminar.
5. En el menú desplegable Acciones, elija Eliminar.

AWS CLI

Eliminar una plantilla de creación de repositorios (AWS CLI)

- Utilice el comando delete-repository-creation-template.html para eliminar una plantilla de creación de repositorios existente. Debe especificar el valor `prefix` de la plantilla. En el siguiente ejemplo, se elimina una plantilla de creación de repositorios con el prefijo `prod`.

  ```
  aws ecr delete-repository-creation-template \
       --prefix prod
  ```

  El resultado del comando mostrará los detalles de la plantilla de creación de repositorios eliminada.

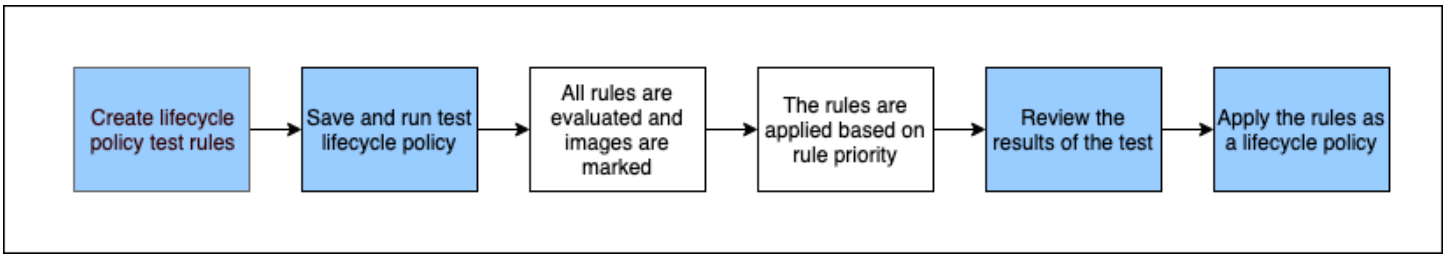
Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR

Las políticas de ciclo de vida de Amazon ECR proporcionan un mayor control sobre la administración del ciclo de vida de las imágenes en un repositorio privado. Una política de ciclo de vida incluye una o varias reglas y cada una de ellas define una acción de Amazon ECR. Según los criterios de caducidad de la política de ciclo de vida, las imágenes caducan en función de su antigüedad o recuento dentro de un plazo de 24 horas. Cuando Amazon ECR realiza una acción basada en una política de ciclo de vida, esta acción se captura como un evento en AWS CloudTrail. Para obtener más información, consulte Registrar las acciones de Amazon ECR con AWS CloudTrail.

Funcionamiento de las políticas de ciclo de vida

Una política de ciclo de vida se compone de una o varias reglas que determinan las imágenes de un repositorio que se deben marcar para vencimiento. Al plantearse el uso de políticas de ciclo de vida, es importante utilizar la vista previa de las políticas de ciclo de vida para confirmar qué imágenes marcará la política de ciclo de vida para vencimiento antes de aplicarla a un repositorio. Una vez que se aplica una política de ciclo de vida a un repositorio, lo normal es que las imágenes caduquen en un plazo de 24 horas tras haber cumplido con los criterios de caducidad. Cuando Amazon ECR realiza una acción basada en una política de ciclo de vida, se captura como un evento en AWS CloudTrail. Para obtener más información, consulte Registrar las acciones de Amazon ECR con AWS CloudTrail.

En el siguiente diagrama se muestra el flujo de trabajo de una política de ciclo de vida.



1. Cree una o varias reglas de prueba.
2. Guarde las reglas de prueba y ejecute la vista previa.
3. El evaluador de políticas de ciclo de vida revisa todas las reglas y marca las imágenes a las que afecta cada regla.

4. A continuación, el evaluador de políticas de ciclo de vida aplica las reglas, en función de la prioridad de la regla, y muestra qué imágenes del repositorio están configuradas para que se marquen para vencimiento. Un número de prioridad de regla más bajo significa una prioridad más alta. Por ejemplo, una regla con prioridad 1 tiene prioridad sobre una regla con prioridad 2.
5. Revise los resultados de la prueba, asegurándose de que las imágenes marcadas para vencimiento sean las previstas.
6. Aplique las reglas de prueba como la política de ciclo de vida del repositorio.
7. Una vez creada una política de ciclo de vida, lo normal es que las imágenes caduquen en un plazo de 24 horas tras haber cumplido con los criterios de caducidad.

Reglas de evaluación de políticas de ciclo de vida

El evaluador de políticas de ciclo de vida se encarga de analizar el código JSON en texto sin formato de la política de ciclo de vida, de evaluar todas las reglas y de aplicarlas en función de la prioridad de regla a las imágenes en el repositorio. A continuación se explica la lógica del evaluador de políticas de ciclo de vida con más detalle. Para ver ejemplos, consulta Ejemplos de políticas de ciclo de vida en Amazon ECR.

- Cuando hay artefactos de referencia en un repositorio, las políticas del ciclo de vida de Amazon ECR los limpian automáticamente en un plazo de 24 horas a partir de la eliminación de la imagen en cuestión.
- Todas las reglas se evalúan al mismo tiempo, independientemente de la prioridad. Después de evaluar todas las reglas, se aplican en función de la prioridad.
- Una imagen puede ser marcada para vencimiento exactamente por una o ninguna regla.
- Una imagen que cumpla los requisitos de etiquetado de una regla no puede ser marcada para vencimiento por una regla con una prioridad menor.
- Las reglas no pueden marcar nunca imágenes que están marcadas con reglas de mayor prioridad, pero puede seguir identificándolas si no han caducado.
- El conjunto de reglas debe contener un conjunto único de prefijos de etiqueta.
- Solo una regla puede seleccionar imágenes sin etiquetar.
- Si una lista de manifiestos hace referencia a una imagen, no puede caducar sin que se elimine primero la lista de manifiestos.
- El vencimiento se ordena siempre por `pushed_at_time` y las imágenes más antiguas caducan siempre antes que las más nuevas.

- Una regla de política de ciclo de vida puede especificar `tagPatternList` o `tagPrefixList`, pero no ambas. Sin embargo, una política de ciclo de vida puede contener varias reglas, y las diferentes reglas utilizan listas de patrones y prefijos. Se encuentra una coincidencia de una imagen si todas las etiquetas del valor `tagPatternList` o `tagPrefixList` coinciden con alguna de las etiquetas de la imagen.
- Los parámetros `tagPatternList` o `tagPrefixList` solo se pueden usar si `tagStatus` es `tagged`.
- Cuando se utiliza `tagPatternList`, una imagen se considera que coincide correctamente si coincide con el filtro comodín. Por ejemplo, si se aplica un filtro de `prod*`, coincidirá con las etiquetas de imagen cuyo nombre comience con `prod`, como `prod`, `prod1` o `production-team1`. Del mismo modo, si se aplica un filtro de `*prod*`, coincidirá con las etiquetas de imagen cuyo nombre contenga `prod`, por ejemplo `repo-production` o `prod-team`.

  > **Important**
  >
  > Hay un límite máximo de cuatro caracteres comodines (*) por cadena. Por ejemplo, `["*test*1*2*3", "test*1*2*3*"]` es válido pero `["test*1*2*3*4*5*6"]` no es válido.

- Cuando se utiliza `tagPrefixList`, se encuentra una coincidencia de una imagen si todos los filtros comodín del valor de `tagPrefixList` coinciden con alguna de las etiquetas de la imagen.
- El parámetro `countUnit` solo se utiliza si `countType` es `sinceImagePushed`.
- Con `countType = imageCountMoreThan`, las imágenes se ordenan de las más nuevas a las más antiguas en función de `pushed_at_time` y se marcan para vencimiento todas las imágenes mayores que el recuento especificado.
- Con `countType = sinceImagePushed`, todas las imágenes cuyo `pushed_at_time` sea anterior al número de días especificado basado en `countNumber` se marcan para vencimiento.

Creación de una vista previa de política de ciclo de vida en Amazon ECR

Puede utilizar una vista previa de una política de ciclo de vida para ver el impacto de una política de ciclo de vida en un repositorio de imágenes antes de aplicarla. Se considera una práctica recomendada realizar una vista previa antes de aplicar una política de ciclo de vida a un repositorio.

> **Note**
>
> Si utiliza la replicación de Amazon ECR para crear copias de un repositorio en distintas regiones o cuentas, tenga en cuenta que una política de ciclo de vida solo puede tomar medidas en los repositorios de la región en la que se creó. Por lo tanto, si tiene activada la replicación, puede considerar la posibilidad de crear una política de ciclo de vida en cada región y cuenta en la que vaya a replicar sus repositorios.

Creación de una vista previa de política de ciclo de vida (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, seleccione la región que contiene el repositorio en el que desea obtener la vista previa de una política de ciclo de vida.
3. En el panel de navegación, en Registro privado, elija Repositorios.
4. En la página Repositorios privados, seleccione un repositorio y luego utilice el menú desplegable Acciones para elegir las políticas del ciclo de vida.
5. En la página de las reglas de la política de ciclo de vida del repositorio, elija Editar reglas de prueba, Crear regla.
6. Especifique los siguientes detalles para cada regla de la política de ciclo de vida.

    a. En Rule priority (Prioridad de la regla), escriba un número para la prioridad de la regla. La prioridad de la regla determina el orden en el que se aplican las reglas de la política del ciclo de vida. Un número más bajo significa una prioridad más alta. Por ejemplo, una regla con prioridad 1 tiene prioridad sobre una regla con prioridad 2.

    b. En Rule description (Descripción de la regla), escriba una descripción para la regla de política de ciclo de vida.

    c. En el estado de la imagen, elija Etiquetada (coincidencia de comodines), Etiquetada (coincidencia de prefijos), Sin etiquetar o Cualquiera.

    > **Important**
    >
    > Si especifica varias etiquetas, solo se seleccionan las imágenes con todas las etiquetas especificadas.

d. Si selecciona Etiquetado (coincidencia de comodines) para el estado de la imagen y, a continuación, en Especificar etiquetas para la coincidencia de comodines, puede especificar una lista de etiquetas de imagen con un comodín (*) sobre las que ejecutar una acción según su política de ciclo de vida. Por ejemplo, si las imágenes están etiquetadas como `prod`, `prod1`, `prod2`, etc., debería especificar `prod*` para ejecutar una acción en todas ellas. Si especifica varias etiquetas, solo se seleccionan las imágenes con todas las etiquetas especificadas.

> **Important**
>
> Hay un límite máximo de cuatro caracteres comodines (*) por cadena. Por ejemplo, `["*test*1*2*3", "test*1*2*3*"]` es válido pero `["test*1*2*3*4*5*6"]` no es válido.

e. Si selecciona Etiquetado (coincidencia de prefijos) para el estado de la imagen y, a continuación, en Especificar etiquetas para la coincidencia de prefijos, puede especificar una lista de etiquetas de imagen para ejecutar una acción según su política de ciclo de vida.

f. En los criterios de coincidencia, seleccione Desde que se envió la imagen o Número de imágenes superior ay, a continuación, especifique un valor.

g. Seleccione Save.

7. Cree reglas de política de ciclo de vida de prueba adicionales repitiendo los pasos 5–7.
8. Para ejecutar la vista previa de la política de ciclo de vida, elija Save and run test (Guardar y ejecutar prueba).
9. En Image matches for test lifecycle rules (Coincidencias de imagen para reglas de ciclo de vida de prueba), revise el impacto de la vista previa de la política de ciclo de vida.
10. Si está conforme con los resultados de la vista preliminar, elija Apply as lifecycle policy para crear una política de ciclo de vida con las reglas especificadas. Una vez aplicada una política de ciclo de vida, las imágenes a las que afecta esta política vencen en un plazo de 24 horas.
11. Si no le convencen los resultados de la vista previa, puede eliminar una o más reglas del ciclo de vida de las pruebas y crear una o más reglas para reemplazarlas y, a continuación, repetir la prueba.

Creación de una política de ciclo de vida para un repositorio en Amazon ECR

Utilice una política de ciclo de vida para crear un conjunto de reglas que marquen las imágenes del repositorio sin utilizar para vencimiento. Una vez que se crea una política de ciclo de vida, las imágenes a las que afecta esta política caducan al cabo de 24 horas.

> **Note**
>
> Si utiliza la replicación de Amazon ECR para crear copias de un repositorio en distintas regiones o cuentas, tenga en cuenta que una política de ciclo de vida solo puede tomar medidas en los repositorios de la región en la que se creó. Por lo tanto, si tiene activada la replicación, puede considerar la posibilidad de crear una política de ciclo de vida en cada región y cuenta en la que vaya a replicar sus repositorios.

Requisito previo

Práctica recomendada: crear una vista previa de la política del ciclo de vida para verificar que las imágenes que hayan expirado a causa de las reglas de la política del ciclo de vida sean las que desea. Para obtener instrucciones, consulte Creación de una vista previa de política de ciclo de vida en Amazon ECR.

Creación de una política de ciclo de vida (AWS Management Console)

1. Abra la consola Amazon ECR en los https://console.aws.amazon.com/ecr/repositorios.
2. En la barra de navegación, seleccione la región que contiene el repositorio para el que desea crear una política de ciclo de vida.
3. En el panel de navegación, en Registro privado, elija Repositorios.
4. En la página Repositorios privados, seleccione un repositorio y luego utilice el menú desplegable Acciones para elegir las políticas del ciclo de vida.
5. En la página de política de ciclo de vida del repositorio, elija Crear regla.
6. Especifique los siguientes detalles para la regla de la política de ciclo de vida.

    a. En Rule priority (Prioridad de la regla), escriba un número para la prioridad de la regla. La prioridad de la regla determina el orden en el que se aplican las reglas de la política del

ciclo de vida. Un número de prioridad de regla más bajo significa una prioridad más alta. Por ejemplo, una regla con prioridad 1 tiene prioridad sobre una regla con prioridad 2.

b. En Rule description (Descripción de la regla), escriba una descripción para la regla de política de ciclo de vida.

c. En el estado de la imagen, elija Etiquetada (coincidencia de comodines), Etiquetada (coincidencia de prefijos), Sin etiquetar o Cualquiera.

> **Important**
>
> Si especifica varias etiquetas, solo se seleccionan las imágenes con todas las etiquetas especificadas.

d. Si selecciona Etiquetado (coincidencia de comodines) para el estado de la imagen y, a continuación, en Especificar etiquetas para la coincidencia de comodines, puede especificar una lista de etiquetas de imagen con un comodín (*) sobre las que ejecutar una acción según su política de ciclo de vida. Por ejemplo, si las imágenes están etiquetadas como `prod`, `prod1`, `prod2`, etc., debería especificar `prod*` para ejecutar una acción en todas ellas. Si especifica varias etiquetas, solo se seleccionan las imágenes con todas las etiquetas especificadas.

> **Important**
>
> Hay un límite máximo de cuatro caracteres comodines (*) por cadena. Por ejemplo, `["*test*1*2*3", "test*1*2*3*"]` es válido pero `["test*1*2*3*4*5*6"]` no es válido.

e. Si selecciona Etiquetado (coincidencia de prefijos) para el estado de la imagen y, a continuación, en Especificar etiquetas para la coincidencia de prefijos, puede especificar una lista de etiquetas de imagen sobre las que actuar según su política de ciclo de vida.

f. En los criterios de coincidencia, seleccione Desde que se envió la imagen o Número de imágenes superior ay, a continuación, especifique un valor.

g. Seleccione Save.

7. Cree reglas de política de ciclo de vida adicionales repitiendo los pasos 5–7.

Creación de una política de ciclo de vida (AWS CLI)

1. Obtenga el nombre del repositorio para el que va a crear la política de ciclo de vida.

   ```
   aws ecr describe-repositories
   ```

2. Cree un archivo local denominado `policy.json` con el contenido de la política de ciclo de vida. Para ver ejemplos de política de ciclo de vida, consulte Ejemplos de políticas de ciclo de vida en Amazon ECR.
3. Cree una política de ciclo de vida especificando el nombre del repositorio y haga referencia al archivo JSON de la política de ciclo de vida que ha creado.

   ```
   aws ecr put-lifecycle-policy \
         --repository-name repository-name \
         --lifecycle-policy-text file://policy.json
   ```

Ejemplos de políticas de ciclo de vida en Amazon ECR

A continuación, se muestran ejemplos de políticas de ciclo de vida con su sintaxis.

Para obtener más información acerca de las propiedades de las políticas, consulte Propiedades de la política de ciclo de vida en Amazon ECR. Para obtener instrucciones sobre cómo crear una política de ciclo de vida mediante el AWS CLI, consulteCreación de una política de ciclo de vida (AWS CLI).

Plantilla de política de ciclo de vida

El contenido de la política de ciclo de vida se evalúa antes de asociarse a un repositorio. A continuación, se muestra la plantilla de sintaxis JSON de la política de ciclo de vida.

```
{
        "rules": [
            {
                "rulePriority": integer,
                "description": "string",
                "selection": {
                    "tagStatus": "tagged"|"untagged"|"any",
                    "tagPatternList": list<string>,
                    "tagPrefixList": list<string>,
                    "countType": "imageCountMoreThan"|"sinceImagePushed",
                    "countUnit": "string",
```

```
                "countNumber": integer
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtrar por antigüedad de las imágenes

En el siguiente ejemplo, se muestra la sintaxis de la política de ciclo de vida de una política que crea el vencimiento de las imágenes con una etiqueta que comienza con `prod` al usar una `tagPatternList` de `prod*` que también tienen más de `14` días.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtrar por el número de imágenes

En el siguiente ejemplo se muestra la sintaxis de una política de ciclo de vida que conserva solo una imagen sin etiquetar y marca para vencimiento todas las demás:

```
{
    "rules": [
        {
```

```
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtrar por varias reglas

El siguiente ejemplo usa varias reglas en una política de ciclo de vida. Se ofrece un repositorio de ejemplo y una política de ciclo de vida con una explicación del resultado.

Ejemplo A

Contenido del repositorio:

- Imagen A, lista de etiquetas: ["beta-1", "prod-1"], insertada: hace 10 días
- Imagen B, lista de etiquetas: ["beta-2", "prod-2"], insertada: hace 9 días
- Imagen C, lista de etiquetas: ["beta-3"], insertada: hace 8 días

Texto de la política de ciclo de vida:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
```

```
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La lógica de esta política de vida sería:

- La regla 1 identifica las imágenes etiquetadas con el prefijo `prod`. Debería marcar las imágenes, empezando por la más antigua, hasta que queden una o menos imágenes. Marca la imagen A para vencimiento.
- La regla 2 identifica las imágenes etiquetadas con el prefijo `beta`. Debería marcar las imágenes, empezando por la más antigua, hasta que queden una o menos imágenes. Marca la imagen A y la imagen B para vencimiento. Sin embargo, la imagen A ya ha sido identificada por la regla 1 y, si la imagen B se marcara para vencimiento, se infringiría la regla 1, por lo que se omite.
- Resultado: la imagen A se marca para vencimiento

Ejemplo B

Este es el mismo repositorio que el del ejemplo anterior, pero se ha cambiado el orden de prioridad de las reglas para ilustrar el resultado.

Contenido del repositorio:

- Imagen A, lista de etiquetas: ["beta-1", "prod-1"], insertada: hace 10 días
- Imagen B, lista de etiquetas: ["beta-2", "prod-2"], insertada: hace 9 días
- Imagen C, lista de etiquetas: ["beta-3"], insertada: hace 8 días

Texto de la política de ciclo de vida:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La lógica de esta política de vida sería:

- La regla 1 identifica las imágenes etiquetadas con el prefijo `beta`. Debería marcar las imágenes, empezando por la más antigua, hasta que queden una o menos imágenes. Se identifican las tres imágenes, y la imagen A y la imagen B se marcan para vencimiento.
- La regla 2 identifica las imágenes etiquetadas con el prefijo `prod`. Debería marcar las imágenes, empezando por la más antigua, hasta que queden una o menos imágenes. No se identifica ninguna imagen porque todas las imágenes disponibles ya han sido identificadas por la regla 1, por lo que no se marcan imágenes adicionales.

- Resultado: la imagen A y la imagen B se marcan para vencimiento.

Filtrar por varias etiquetas en una sola regla

Los siguientes ejemplos especifican la sintaxis de la política de ciclo de vida para varios patrones de etiqueta en una sola regla. Se ofrece un repositorio de ejemplo y una política de ciclo de vida con una explicación del resultado.

Ejemplo A

Cuando se especifican varios patrones de etiqueta en una sola regla, las imágenes deben coincidir con todos los patrones de etiqueta especificados.

Contenido del repositorio:

- Imagen A, lista de etiquetas: ["alpha-1"], insertada: hace 12 días
- Imagen B, lista de etiquetas: ["beta-1"], insertada: hace 11 días
- Imagen C, lista de etiquetas: ["alpha-2", "beta-2"], insertada: hace 10 días
- Imagen D, lista de etiquetas: ["alpha-3"], insertada: hace 4 días
- Imagen E, lista de etiquetas: ["beta-3"], insertada: hace 3 días
- Imagen F, lista de etiquetas: ["alpha-4", "beta-4"], insertada: hace 2 días

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
            },
            "action": {
                "type": "expire"
            }
        }
```

```
    ]
}
```

La lógica de esta política de vida sería:

- La regla 1 identifica las imágenes etiquetadas con el prefijo `alpha` y `beta`. Se identifican las imágenes C y F y se marcan las imágenes que tienen más de cinco días (la imagen C).
- Resultado: la imagen C se marca para vencimiento

Ejemplo B

El ejemplo siguiente ilustra que las etiquetas no son exclusivas.

Contenido del repositorio:

- Imagen A, lista de etiquetas: ["alpha-1", "beta-1", "gamma-1"], insertada: hace 10 días
- Imagen B, lista de etiquetas: ["alpha-2", "beta-2"], insertada: hace 9 días
- Imagen C, lista de etiquetas: ["alpha-3", "beta-3", "gamma-2"], insertada: hace 8 días

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La lógica de esta política de vida sería:

- La regla 1 identifica las imágenes etiquetadas con el prefijo `alpha` y `beta`. Se identifican todas las imágenes. Debería marcar las imágenes, empezando por la más antigua, hasta que queden una o menos imágenes. Se marcan la imagen A y la imagen B para vencimiento.
- Resultado: la imagen A y la imagen B se marcan para vencimiento.

Filtrar por todas las imágenes

Los siguientes ejemplos de políticas de ciclo de vida especifican todas las imágenes con filtros distintos. Se ofrece un repositorio de ejemplo y una política de ciclo de vida con una explicación del resultado.

Ejemplo A

A continuación, se muestra la sintaxis de una política de ciclo de vida que se aplica a todas las reglas propiedad conserva solo una imagen sin etiquetar y marca para vencimiento todas las demás.

Contenido del repositorio:

- Imagen A, lista de etiquetas: ["alpha-1"], insertada: hace 4 días
- Imagen B, lista de etiquetas: ["beta-1"], insertada: hace 3 días
- Imagen C, lista de etiquetas: [], insertada: hace 2 días
- Imagen D, lista de etiquetas: ["alpha-2"], insertada: hace 1 día

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
```

```
}
```

La lógica de esta política de vida sería:

- La regla 1 identifica todas las imágenes. Detecta las imágenes A, B, C y D. Debe marcar para vencimiento todas las imágenes excepto la más reciente. Marca las imágenes A, B y C para vencimiento.
- Resultado: las imágenes A, B y C están vencidas.

Ejemplo B

El siguiente ejemplo ilustra una política de ciclo de vida que combina todos los tipos de reglas en una sola política.

Contenido del repositorio:

- Imagen A, lista de etiquetas: ["alpha-1", "beta-1"], insertada: hace 4 días
- Imagen B, lista de etiquetas: [], insertada: hace 3 días
- Imagen C, lista de etiquetas: ["alpha-2"], insertada: hace 2 días
- Imagen D, lista de etiquetas: ["git hash"], insertada: hace 1 día
- Imagen E, lista de etiquetas: [], insertada: hace 1 día

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
```

```
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 3,
            "description": "Rule 3",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

La lógica de esta política de vida sería:

- La regla 1 identifica las imágenes etiquetadas con el prefijo `alpha`. Identifica las imágenes A y C. Debe mantener la imagen más reciente y marcar el resto para vencimiento. Marca la imagen A para vencimiento.
- La regla 2 identifica las imágenes sin etiquetar. Identifica las imágenes B y E. Debe marcar todas las imágenes anteriores a un día para vencimiento. Marca la imagen B para vencimiento.
- La regla 3 identifica todas las imágenes. Identifica las imágenes A, B, C, D y E. Debe mantener la imagen más reciente y marcar el resto para vencimiento. Sin embargo, no puede marcar las imágenes A, B, C o E porque se identificaron con reglas de mayor prioridad. Marca la imagen D para vencimiento.
- Resultado: las imágenes A, B y D están vencidas.

Propiedades de la política de ciclo de vida en Amazon ECR

Las políticas de ciclo de vida tienen las siguientes propiedades.

Para ver ejemplos de políticas de ciclo de vida, consulte Ejemplos de políticas de ciclo de vida en Amazon ECR. Para obtener instrucciones sobre cómo crear una política de ciclo de vida mediante el AWS CLI, consulteCreación de una política de ciclo de vida (AWS CLI).

Prioridad de las reglas

`rulePriority`

Tipo: número entero

Obligatorio: sí

Establece el orden en el que se aplican las reglas, de menor a mayor. Primero se aplica una regla de política de ciclo de vida con una prioridad de 1; después, una regla con una prioridad de 2, y así sucesivamente. Cuando añade reglas a una política de ciclo de vida, debe asignar a cada una un valor único para `rulePriority`. Los valores en las reglas de una política no necesitan ser secuenciales. Una regla con un valor `tagStatus` de `any` debe tener el valor más alto para `rulePriority` y evaluarse en último lugar.

Descripción

`description`

Tipo: string

Obligatorio: no

(Opcional) Describe la finalidad de una regla de una política de ciclo de vida.

Estado de etiqueta

`tagStatus`

Tipo: cadena

Obligatorio: sí

Determina si la regla de la política de ciclo de vida que añade especifica una etiqueta para una imagen. Las opciones válidas son `tagged`, `untagged` o `any`. Si especifica `any`, se evalúa la regla en todas las imágenes. Si especifica `tagged`, también debe especificar valor de `tagPrefixList`. Si especifica `untagged`, también debe omitir `tagPrefixList`.

Lista de patrones de etiquetas

`tagPatternList`

Tipo: lista[cadena]

Obligatorio: sí, en caso de que `tagStatus` esté configurado como etiquetado y `tagPrefixList` no está especificado

Al crear una política de ciclo de vida para las imágenes etiquetadas, se recomienda utilizar una `tagPatternList` para especificar cuáles son las etiquetas que van a caducar. Debe especificar una lista separada por comas de patrones de etiquetas de imagen que pueden contener comodines (`*`) en los que se va a realizar una acción con la política de ciclo de vida. Por ejemplo, si las imágenes están etiquetadas como `prod`, `prod1`, `prod2`, etc., debería usar la lista del patrón de etiqueta `prod*` para especificarlas todas. Si especifica varias etiquetas, solo se seleccionan las imágenes con todas las etiquetas especificadas.

> **Important**
>
> Hay un límite máximo de cuatro caracteres comodines (`*`) por cadena. Por ejemplo, `["*test*1*2*3", "test*1*2*3*"]` es válido pero `["test*1*2*3*4*5*6"]` no es válido.

Lista de prefijos de etiqueta

`tagPrefixList`

Tipo: lista[cadena]

Obligatorio: sí, en caso de que `tagStatus` esté configurado como etiquetado y `tagPatternList` no está especificado

Solo se usa si especificó `"tagStatus": "tagged"` y no está especificando un `tagPatternList`. Debe especificar una lista separada por comas de prefijos de etiquetas de imagen en los que se va a realizar una acción con la política de ciclo de vida. Por ejemplo, si las imágenes están etiquetadas como `prod`, `prod1`, `prod2`, etc., debería usar el prefijo de etiqueta `prod` para especificarlas todas. Si especifica varias etiquetas, solo se seleccionan las imágenes con todas las etiquetas especificadas.

Tipo de recuento

`countType`

Tipo: cadena

Obligatorio: sí

Especifique un tipo de recuento que desea aplicar a las imágenes.

Si `countType` está establecido en `imageCountMoreThan`, también especifica `countNumber` para crear una regla que establezca el límite de imágenes que hay en el repositorio. Si `countType` está establecido en `sinceImagePushed`, también especifica `countUnit` y `countNumber` para definir un límite de tiempo de las imágenes que hay en el repositorio.

Unidad de recuento

`countUnit`

Tipo: cadena

Obligatorio: sí, solo si `countType` está establecido en `sinceImagePushed`

Especifique una unidad de recuento de `days` para indicar la unidad de tiempo, además de `countNumber`, que es el número de días.

Solo debe especificarse cuando `countType` sea `sinceImagePushed`; se producirá un error si especifica una unidad de recuento cuando `countType` es otro valor.

Cantidad

`countNumber`

Tipo: número entero

Obligatorio: sí

Especifique un número de recuento. Los valores aceptables son enteros positivos (el valor `0` no se acepta).

Si el `countType` usado es `imageCountMoreThan`, el valor es el número máximo de imágenes que puede conservar en el repositorio. Si el `countType` usado es `sinceImagePushed`, el valor es el límite de antigüedad máxima de las imágenes.

Acción

`type`

Tipo: string

Obligatorio: sí

Especifique un tipo de acción. El valor admitido es `expire`.

Seguridad en Amazon Elastic Container Registry

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores independientes prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información sobre los programas de conformidad que se aplican a Amazon ECR, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También eres responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación lo ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon ECR. En los siguientes temas, se mostrará cómo configurar Amazon ECR para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger sus recursos de Amazon ECR.

Temas

- Identity and Access Management para Amazon Elastic Container Registry
- Protección de los datos en Amazon ECR
- Validación de conformidad para Amazon Elastic Container Registry
- Seguridad de la infraestructura en Amazon Elastic Container Registry
- Prevención de la sustitución confusa entre servicios

Identity and Access Management para Amazon Elastic Container Registry

AWS Identity and Access Management (IAM) es un sistema Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de Amazon ECR. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración de acceso mediante políticas
- Cómo funciona Amazon Elastic Container Registry con IAM
- Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry
- Uso del control de acceso basado en etiquetas
- AWS políticas gestionadas para Amazon Elastic Container Registry
- Uso de roles vinculados a servicios para Amazon ECR
- Solución de problemas de identidad y acceso para Amazon Elastic Container Registry

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que realice en Amazon ECR.

Usuario de servicio: si utiliza el servicio Amazon ECR para realizar su trabajo, su administrador proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Amazon ECR para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos al administrador. Si no puede acceder a una característica de Amazon ECR, consulte Solución de problemas de identidad y acceso para Amazon Elastic Container Registry.

Administrador de servicio: si está a cargo de los recursos de Amazon ECR de su empresa, probablemente disponga de acceso completo a Amazon ECR. Su trabajo consiste en determinar

a qué características y recursos de Amazon ECR deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo la empresa puede utilizar IAM con Amazon ECR, consulte Cómo funciona Amazon Elastic Container Registry con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que desee obtener detalles sobre cómo escribir políticas para administrar el acceso a Amazon ECR. Para consultar ejemplos de políticas de Amazon ECR basadas en identidades que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (IAM Identity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su gestionador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre la firma de solicitudes, consulte AWS Signature Versión 4 para solicitudes API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Autenticación multifactor AWS en IAM en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utiliza el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puedes iniciar sesión como grupo. Puedes usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una persona determinada. Para asumir temporalmente un rol de IAM en el AWS Management Console, puede cambiar de un rol de usuario

a uno de IAM (consola). Puedes asumir un rol llamando a una operación de AWS API AWS CLI o usando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulta Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles de federación, consulte Crear un rol para un proveedor de identidad de terceros (federación) en la Guía de usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulta Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulta Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en AWS ellas, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para

obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

- Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y realizan AWS CLI solicitudes a la AWS API. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte Usar un rol de IAM para conceder permisos a las aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utiliza para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para obtener más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puedes establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): SCPs son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una Usuario raíz de la cuenta de AWS. Para obtener más información sobre Organizations SCPs, consulte las políticas de control de servicios en la Guía del AWS Organizations usuario.
- Políticas de control de recursos (RCPs): RCPs son políticas de JSON que puedes usar para establecer los permisos máximos disponibles para los recursos de tus cuentas sin actualizar las políticas de IAM asociadas a cada recurso que poseas. El RCP limita los permisos de los recursos en las cuentas de los miembros y puede afectar a los permisos efectivos de las identidades, incluidos los permisos Usuario raíz de la cuenta de AWS, independientemente de si pertenecen a su organización. Para obtener más información sobre Organizations e RCPs incluir una lista de Servicios de AWS ese apoyo RCPs, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud

cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon Elastic Container Registry con IAM

Antes de utilizar IAM para administrar el acceso a Amazon ECR, debe conocer qué características de IAM se encuentran disponibles con Amazon ECR. Para obtener una visión general de cómo Amazon ECR y otros AWS servicios funcionan con IAM, consulte AWS Servicios que funcionan con IAM en la Guía del usuario de IAM.

Temas

- Políticas de Amazon ECR basadas en identidades
- Políticas basadas en recursos de Amazon ECR
- Autorización basada en etiquetas de Amazon ECR
- Roles de IAM de Amazon ECR

Políticas de Amazon ECR basadas en identidades

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Amazon ECR admite acciones, claves de condiciones y recursos específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones de políticas de Amazon ECR utilizan el siguiente prefijo antes de la acción: `ecr:`. Por ejemplo, para conceder a alguien permiso para crear un repositorio de Amazon

ECR con la operación `CreateRepository` de la API de Amazon ECR, incluya la acción `ecr:CreateRepository` en su política. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. Amazon ECR define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "ecr:Describe*"
```

Para ver una lista de las acciones de Amazon ECR, consulte Acciones, recursos y claves de condiciones de Amazon Elastic Container Registry en la Guía del usuario de IAM.

Recursos

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puedes hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utiliza un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Un recurso de repositorio de Amazon ECR tiene el siguiente ARN:

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

Para obtener más información sobre el formato de ARNs, consulte Amazon Resource Names (ARNs) y AWS Service Namespaces.

Por ejemplo, para especificar el repositorio `my-repo` en la región `us-east-1` en la instrucción, utilice el siguiente ARN:

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

Para especificar todos los repositorios que pertenecen a una cuenta específica, utilice el carácter comodín (*):

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

Para especificar varios recursos en una sola sentencia, sepárelos ARNs con comas.

```
"Resource": [
      "resource1",
      "resource2"
```

Para ver una lista de los tipos de recursos de Amazon ECR y sus respectivos tipos ARNs, consulte Recursos definidos por Amazon Elastic Container Registry en la Guía del usuario de IAM. Para obtener información acerca de las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por Amazon Elastic Container Registry.

Claves de condición

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puedes crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación

lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puedes utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puedes conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulta Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Amazon ECR define su propio conjunto de claves de condición y también admite el uso de algunas claves de condición globales. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

La mayoría de las acciones de Amazon ECR admiten las claves de condición `aws:ResourceTag` y `ecr:ResourceTag`. Para obtener más información, consulte Uso del control de acceso basado en etiquetas.

Para ver una lista de las claves de condición de Amazon ECR, consulte Claves de condición definidas por Amazon Elastic Container Registry en la Guía del usuario de IAM. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por Amazon Elastic Container Registry.

Ejemplos

Para ver ejemplos de políticas basadas en identidad de Amazon ECR, consulte Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry.

Políticas basadas en recursos de Amazon ECR

Las políticas basadas en recursos son documentos de políticas JSON que especifican qué acciones puede realizar una entidad de servicio especificada en un recurso de Amazon ECR y en qué condiciones. Amazon ECR admite políticas de permisos basadas en recursos para repositorios de Amazon ECR. Las políticas basadas en recursos le permiten conceder a otras cuentas de permisos de uso para cada recurso. También puede utilizar una política basada en recursos para permitir a un servicio de AWS acceder a los repositorios de Amazon ECR.

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política basada en recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Si el principal y el recurso están en AWS cuentas diferentes, también debe conceder permiso a la entidad principal para acceder al recurso. Conceda permiso asociando a la entidad una política basada en identidades. Sin embargo, si una política basada en recursos concede acceso a un principal de la misma cuenta, no necesitará permisos adicionales para el repositorio de Amazon ECR en la política basada en la identidad. Para obtener más información, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

El servicio de Amazon ECR solo admite un tipo de política basada en recursos denominada política de repositorios, que se adjunta a un repositorio. Esta política indica las entidades principales (cuentas, usuarios, roles y usuarios federados) que pueden realizar acciones en el repositorio. Para obtener información sobre cómo asociar una política basada en recursos a un repositorio, consulte Políticas de repositorios privados en Amazon ECR.

> **Note**
>
> En una política de repositorio de Amazon ECR, el elemento de política `Sid` admite caracteres y espacios adicionales que no se admiten en las políticas de IAM.

Ejemplos

Para ver ejemplos de políticas basadas en recursos de Amazon ECR, consulte Ejemplos de políticas de repositorio privado en Amazon ECR.

Autorización basada en etiquetas de Amazon ECR

Puede adjuntar etiquetas a los recursos de Amazon ECR o transferirlas en una solicitud a Amazon ECR. Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `ecr:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Para obtener más información sobre el etiquetado de recursos de Amazon ECR, consulte Etiquetado de un repositorio privado en Amazon ECR.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Uso del control de acceso basado en etiquetas.

Roles de IAM de Amazon ECR

Un rol de IAM es una entidad de su cuenta que tiene permisos específicos. AWS

Uso de credenciales temporales con Amazon ECR

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Para obtener credenciales de seguridad temporales, puede llamar a operaciones de AWS STS API como AssumeRoleo GetFederationToken.

Amazon ECR admite el uso de credenciales temporales.

Roles vinculados a servicios

Los roles vinculados a un servicio permiten a AWS los servicios acceder a los recursos de otros servicios para completar una acción en tu nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Amazon ECR admite roles vinculados a servicios. Para obtener más información, consulte Uso de roles vinculados a servicios para Amazon ECR.

Ejemplos de políticas basadas en identidad de Amazon Elastic Container Registry

De manera predeterminada, los usuarios y roles no tienen permiso para crear ni modificar recursos de Amazon ECR. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS la API. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Amazon ECR, incluido el formato de cada uno de los tipos de recursos, consulte Acciones, recursos y claves de condición de Amazon Elastic Container Registry en la Referencia de autorización de servicios. ARNs

Para obtener información acerca de cómo crear una política basada en identidad de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Temas

- Prácticas recomendadas relativas a políticas
- Uso de la consola de Amazon ECR
- Permitir a los usuarios ver sus propios permisos
- Acceso a un repositorio de Amazon ECR

Prácticas recomendadas relativas a políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de Amazon ECR de su cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulta las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulta Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulta Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de Amazon ECR

Para acceder a la consola de Amazon Elastic Container Registry, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de Amazon ECR de su AWS cuenta. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

Para garantizar que esas entidades puedan seguir utilizando la consola de Amazon ECR, añada la política `AmazonEC2ContainerRegistryReadOnly` AWS gestionada a las entidades. Para obtener más información, consulte Agregar de permisos a un usuario en la Guía del usuario de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
```

```
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
                "ecr:DescribeImageScanFindings"
            ],
            "Resource": "*"
        }
    ]
}
```

No es necesario conceder permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intenta realizar.

Permitir a los usuarios ver sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas gestionadas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Acceso a un repositorio de Amazon ECR

En este ejemplo, quiere conceder a un usuario de su AWS cuenta acceso a uno de sus repositorios de Amazon ECR,. `my-repo` También desea permitir que el usuario inserte, extraiga y enumere imágenes.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"GetAuthorizationToken",
         "Effect":"Allow",
         "Action":[
            "ecr:GetAuthorizationToken"
         ],
         "Resource":"*"
      },
      {
         "Sid":"ManageRepositoryContents",
         "Effect":"Allow",
         "Action":[
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:InitiateLayerUpload",
```

```
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource":"arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
        }
    ]
}
```

Uso del control de acceso basado en etiquetas

La acción de la `CreateRepository` API Amazon ECR le permite especificar etiquetas al crear el repositorio. Para obtener más información, consulte Etiquetado de un repositorio privado en Amazon ECR.

Para que los usuarios puedan etiquetar repositorios durante la creación, es preciso que tengan permisos para utilizar la acción que crea el recurso (por ejemplo, `ecr:CreateRepository`). Si se especifican etiquetas en la acción de creación de recursos, Amazon realiza una autorización adicional en la acción `ecr:CreateRepository` para verificar que los usuarios tengan permisos para crear etiquetas.

Puede utilizar el control de acceso basado en etiquetas con las políticas de IAM. A continuación se muestran algunos ejemplos.

La siguiente política solo permitiría a un usuario crear o etiquetar un repositorio como `key=environment,value=dev`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateTaggedRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
```

```
        },
        {
            "Sid": "AllowTagRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        }
    ]
}
```

La siguiente política permitiría a un usuario extraer imágenes de todos los repositorios, a menos que estén etiquetadas como tal. `key=environment,value=prod`

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/environment": "prod"
                }
            }
        }
```

```
    ]
}
```

AWS políticas gestionadas para Amazon Elastic Container Registry

Una política AWS gestionada es una política independiente creada y administrada por. AWS AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Amazon ECR proporciona varias políticas gestionadas que puede adjuntar a las identidades de IAM o a las instancias de Amazon EC2 . Estas políticas administradas permiten diferentes niveles de control sobre el acceso a los recursos y a las operaciones de API de Amazon ECR. Para obtener más información acerca de cada una de las operaciones de API mencionadas en estas políticas, consulte Acciones en la Referencia de la API de Amazon Elastic Container Registry.

Temas

- AmazonEC2ContainerRegistryFullAccess
- AmazonEC2ContainerRegistryPowerUser
- AmazonEC2ContainerRegistryPullOnly
- AmazonEC2ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- ECRTemplateServiceRolePolicy
- Amazon ECR actualiza las políticas AWS gestionadas

AmazonEC2ContainerRegistryFullAccess

Puede adjuntar la política `AmazonEC2ContainerRegistryFullAccess` a las identidades de IAM.

Puede usar esta política administrada como punto de partida para crear su propia política de IAM en función de sus requisitos específicos. Por ejemplo, puede crear una política específica para proporcionar a un usuario o rol acceso completo de administrador con objeto de administrar el uso de Amazon ECR. Con la característica Políticas de ciclo de vida de Amazon ECR, puede especificar la administración del ciclo de vida de las imágenes en un repositorio. Los eventos de las políticas de ciclo de vida se notifican como CloudTrail eventos. Amazon ECR está integrado AWS CloudTrail para que pueda mostrar los eventos de su política de ciclo de vida directamente en la consola de Amazon ECR. La política de IAM administrada `AmazonEC2ContainerRegistryFullAccess` incluye el permiso `cloudtrail:LookupEvents` para facilitar este comportamiento.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `ecr`— Permite a los directores el acceso total a todos los Amazon APIs ECR.
- `cloudtrail`— Permite a los directores buscar los eventos de administración o los eventos de AWS CloudTrail Insights capturados por ellos. CloudTrail

La política `AmazonEC2ContainerRegistryFullAccess` es como sigue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "cloudtrail:LookupEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "replication.ecr.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AmazonEC2ContainerRegistryPowerUser

Puede adjuntar la política `AmazonEC2ContainerRegistryPowerUser` a las identidades de IAM.

Esta política otorga permisos administrativos que permiten que los usuarios de IAM lean y escriban en los repositorios, pero no les permite eliminar los repositorios ni cambiar los documentos de la política aplicados a ellos.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `ecr`: permite a las entidades principales leer y escribir en los repositorios, así como leer políticas de ciclo de vida. Las entidades de servicio no tienen permiso para eliminar repositorios o cambiar las políticas de ciclo de vida que se les aplican.

La política `AmazonEC2ContainerRegistryPowerUser` es como sigue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
```

```
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
                "ecr:DescribeImageScanFindings",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonEC2ContainerRegistryPullOnly

Puede adjuntar la política `AmazonEC2ContainerRegistryPullOnly` a las identidades de IAM.

Esta política da permiso para extraer imágenes de contenedores de Amazon ECR. Si el registro está habilitado para caché de extracción, también permitirá que las extracciones importen una imagen de un registro principal.

Detalles de los permisos

Esta política incluye los siguientes permisos:

- `ecr`: permite a las entidades de seguridad leer repositorios y sus respectivas políticas de ciclo de vida.

La política `AmazonEC2ContainerRegistryPullOnly` es como sigue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchImportUpstreamImage"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonEC2ContainerRegistryReadOnly

Puede adjuntar la política `AmazonEC2ContainerRegistryReadOnly` a las identidades de IAM.

Esta política otorga permisos de solo lectura a Amazon ECR. Esto incluye la capacidad de mostrar una lista de repositorios e imágenes en los repositorios. También incluye la capacidad de extraer imágenes de Amazon ECR con la CLI de Docker.

Detalles de los permisos

Esta política incluye los siguientes permisos:

- `ecr`: permite a las entidades de seguridad leer repositorios y sus respectivas políticas de ciclo de vida.

La política `AmazonEC2ContainerRegistryReadOnly` es como sigue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
                "ecr:DescribeImageScanFindings"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

AWSECRPullThroughCache_ServiceRolePolicy

No puede adjuntar la política de IAM administrada `AWSECRPullThroughCache_ServiceRolePolicy` a sus entidades de IAM. Esta política se adjunta a un rol vinculado a servicios que permite a Amazon ECR enviar imágenes a los repositorios a través del flujo de trabajo de extracción de caché. Para obtener más información, consulte Rol vinculado a servicios de Amazon ECR para la caché de extracción.

ECRReplicationServiceRolePolicy

No puede adjuntar la política de IAM administrada `ECRReplicationServiceRolePolicy` a sus entidades de IAM. Esta política se encuentra adjunta a un rol vinculado a un servicio que permite a Amazon ECR realizar acciones en su nombre. Para obtener más información, consulte Uso de roles vinculados a servicios para Amazon ECR.

ECRTemplateServiceRolePolicy

No puede adjuntar la política de IAM administrada `ECRTemplateServiceRolePolicy` a sus entidades de IAM. Esta política se encuentra adjunta a un rol vinculado a un servicio que permite a Amazon ECR realizar acciones en su nombre. Para obtener más información, consulte Uso de roles vinculados a servicios para Amazon ECR.

Amazon ECR actualiza las políticas AWS gestionadas

Consulte los detalles sobre las actualizaciones de las políticas AWS gestionadas de Amazon ECR desde el momento en que este servicio comenzó a realizar el seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página de historial de documentos de Amazon ECR.

Cambio	Descripción	Fecha
Rol vinculado a servicios de Amazon ECR para la caché	Amazon ECR agregó nuevos permisos a la política `AWSECRPullThroughC`	12 de marzo de 2025

Cambio	Descripción	Fecha
de extracción: actualización de una política actual	`ache_ServiceRolePo licy` . Estos permisos permiten a Amazon ECR extraer imágenes del registro privado de ECR. Esto es obligatorio cuando se utiliza una regla de caché de extracción para almacenar en caché imágenes de otro registro privado de Amazon ECR.	
Amazon EC2 Container RegistryPullOnly — Nueva política	Amazon ECR ha añadido una nueva política que concede permisos de solo extracción a Amazon ECR.	10 de octubre de 2024
ECRTemplateService RolePolicy: política nueva	Amazon ECR agregó una política nueva. Esta política está asociada con el rol vinculado a servicios `ECRTemplateService RolePolicy` para la característica de plantilla de creación de repositorios.	20 de junio de 2024

Cambio	Descripción	Fecha
AWSECRPullThroughCache_ServiceRolePolicy: actualización de una política actual	Amazon ECR agregó nuevos permisos a la política `AWSECRPullThroughCache_ServiceRolePolicy` . Estos permisos permiten a Amazon ECR recuperar el contenido cifrado de un secreto de Secrets Manager. Esto es necesario cuando se utiliza una regla de caché de extracción para almacenar en caché las imágenes de un registro anterior que requiere autenticación.	15 de noviembre de 2023
AWSECRPullThroughCache_ServiceRolePolicy: política nueva	Amazon ECR agregó una política nueva. Esta política está asociada con el rol vinculado a servicios `AWSServiceRoleForECRPullThroughCache` para la característica de extracción de caché.	29 de noviembre de 2021
ECRReplicationServiceRolePolicy: política nueva	Amazon ECR agregó una política nueva. Esta política está asociada con el rol vinculado a servicios `AWSServiceRoleForECRReplication` para la característica de replicación.	4 de diciembre de 2020

Cambio	Descripción	Fecha
Amazon EC2 Container RegistryFullAccess: actualización de una política existente	Amazon ECR agregó nuevos permisos a la política `AmazonEC2ContainerRegistryFullAccess`. Estos permisos permiten que las entidades de seguridad creen el rol vinculado al servicio de Amazon ECR.	4 de diciembre de 2020
Amazon EC2 Container RegistryReadOnly: actualización de una política existente	Amazon ECR agregó nuevos permisos a la política `AmazonEC2ContainerRegistryReadOnly` que permite a las entidades de seguridad leer políticas del ciclo de vida, mostrar una lista de etiquetas y describir los resultados del escaneo para las imágenes.	10 de diciembre de 2019
Amazon EC2 Container RegistryPowerUser: actualización de una política existente	Amazon ECR agregó nuevos permisos a la política `AmazonEC2ContainerRegistryPowerUser`. Estos permiten a las entidades de seguridad leer políticas del ciclo de vida, mostrar una lista de etiquetas y describir los resultados del escaneo para las imágenes.	10 de diciembre de 2019

Cambio	Descripción	Fecha
Amazon EC2 Container RegistryFullAccess: actualización de una política existente	Amazon ECR agregó nuevos permisos a la política `AmazonEC2ContainerRegistryFullAccess`. Permiten a los directores buscar los eventos de administración o los eventos de AWS CloudTrail Insights capturados por CloudTrail.	10 de noviembre de 2017
Amazon EC2 Container RegistryReadOnly: actualización de una política existente	Amazon ECR agregó nuevos permisos a la política `AmazonEC2ContainerRegistryReadOnly`. Permiten a las entidades de seguridad describir imágenes de Amazon ECR.	11 de octubre de 2016
Amazon EC2 Container RegistryPowerUser: actualización de una política existente	Amazon ECR agregó nuevos permisos a la política `AmazonEC2ContainerRegistryPowerUser`. Permiten a las entidades de seguridad describir imágenes de Amazon ECR.	11 de octubre de 2016

Cambio	Descripción	Fecha
Amazon EC2 Container RegistryReadOnly — Nueva política	Amazon ECR ha añadido una nueva política que concede permisos de solo lectura a Amazon ECR. Estos incluyen la capacidad de mostrar una lista de repositorios e imágenes en los repositorios. También incluyen la capacidad de extraer imágenes de Amazon ECR con la CLI de Docker.	21 de diciembre de 2015
Amazon EC2 Container RegistryPowerUser — Nueva política	Amazon ECR agregó una nueva política que otorga permisos administrativos que permiten a los usuarios leer y escribir en los repositorios, pero no les permite eliminar repositorios ni cambiar los documentos de política que se les aplican.	21 de diciembre de 2015
Amazon EC2 Container RegistryFullAccess — Nueva política	Amazon ECR agregó una política nueva. Esta política otorga acceso completo a Amazon ECR.	21 de diciembre de 2015
Amazon ECR comenzó a realizar el seguimiento de los cambios	Amazon ECR comenzó a realizar un seguimiento de los cambios en las políticas AWS gestionadas.	24 de junio de 2021

Uso de roles vinculados a servicios para Amazon ECR

Amazon Elastic Container Registry (Amazon ECR) AWS Identity and Access Management utiliza funciones vinculadas a servicios (IAM) para proporcionar los permisos necesarios para utilizar las funciones de replicación y extracción de memoria caché. Un rol vinculado a un servicio es un tipo único de rol de IAM que se encuentra vinculado directamente a Amazon ECR. Amazon ECR predefine el rol vinculado a un servicio. Incluye todos los permisos que el servicio requiere para respaldar la replicación y las características de caché de su registro privado. Después de configurar la replicación o la extracción de caché de su registro, se crea automáticamente un rol vinculado a servicios en su nombre. Para obtener más información, consulte Configuración del registro privado en Amazon ECR.

Un rol vinculado a servicios facilita la configuración de la replicación y la extracción de caché con Amazon ECR. Esto se debe a que, al usarlo, no tendrá que agregar manualmente todos los permisos necesarios. Amazon ECR define los permisos de sus roles vinculados a servicios y, a menos que esté definido de otra manera, solo Amazon ECR puede asumir sus roles. Los permisos definidos incluyen la política de confianza y la política de permisos. La política de permisos no se puede adjuntar a ninguna otra entidad de IAM.

Puede eliminar el rol vinculado a servicios correspondiente solo después de desactivar la replicación o la extracción de caché en su registro. Esto garantiza que no elimine inadvertidamente los permisos que Amazon ECR requiere para estas características.

Para obtener más información sobre otros servicios que admiten los roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. En esta página vinculada, busque los servicios para los que se indique Sí en la columna Roles vinculados a servicios. Elija una opción Sí con un enlace para ver la documentación relevante acerca del rol vinculado al servicio en cuestión.

Temas

- Regiones admitidas para los roles vinculados a servicios de Amazon ECR
- Rol vinculado a servicios de Amazon ECR para replicación
- Rol vinculado a servicios de Amazon ECR para la caché de extracción
- Rol vinculado a servicios de Amazon ECR para plantillas de creación de repositorios

Regiones admitidas para los roles vinculados a servicios de Amazon ECR

Amazon ECR admite el uso de roles vinculados a servicios en todas las regiones en las que se encuentra disponible el servicio Amazon ECR. Para obtener más información acerca de la disponibilidad de la región de Amazon ECR, consulte Regiones y puntos de conexión de AWS.

Rol vinculado a servicios de Amazon ECR para replicación

Amazon ECR utiliza un rol vinculado a un servicio denominado que AWSServiceRoleForECRReplicationpermite a Amazon ECR replicar imágenes en varias cuentas.

Permisos de roles vinculados a servicios para Amazon ECR

El rol AWSService RoleFor ECRReplication vinculado al servicio confía en que los siguientes servicios asuman el rol:

- `replication.ecr.amazonaws.com`

La siguiente política de permisos del rol `ECRReplicationServiceRolePolicy` permite que Amazon ECR use las siguientes acciones en los recursos:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> `ReplicateImage` es una API interna que Amazon ECR utiliza para la replicación y a la que no se puede llamar directamente.

Debe configurar permisos para permitir a una entidad de IAM (como, por ejemplo, un usuario, grupo o rol) crear, editar o eliminar la descripción de un rol vinculado a un servicio. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio para Amazon ECR

No necesita crear manualmente el rol vinculado al servicio de Amazon ECR. Al configurar los ajustes de replicación para su registro en la AWS Management Console AWS CLI, la o la AWS API, Amazon ECR crea automáticamente la función vinculada al servicio.

Si elimina este rol vinculado a un servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Al configurar los parámetros de replicación para el registro, Amazon ECR crea automáticamente el rol vinculado a un servicio otra vez.

Edición de un rol vinculado a un servicio para Amazon ECR

Amazon ECR no permite editar manualmente el rol vinculado al AWSService RoleFor ECRReplication servicio. Después de crear un rol vinculado a un servicio, no puede cambiarle el nombre, ya que varias entidades pueden hacer referencia a él. Sin embargo, puede editar la descripción del rol mediante IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio para Amazon ECR

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a servicios, recomendamos que elimine dicho rol. De esta forma no conservará una entidad no utilizada que no se monitorice ni se mantenga de forma activa. No obstante, debe quitar la configuración de replicación del registro en cada región para poder eliminar manualmente el rol vinculado a un servicio.

> **Note**
>
> Si intenta eliminar recursos mientras el servicio Amazon ECR sigue utilizando los roles, es posible que la acción de eliminación falle. Si eso sucede, espere unos minutos e inténtelo de nuevo.

Para eliminar los recursos de Amazon ECR utilizados por el AWSService RoleFor ECRReplication

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.

2. En la barra de navegación, elija la región en la que está establecida la configuración de replicación.
3. En el panel de navegación, elija Private registry (Registro privado).
4. En la página Private registry (Registro privado), en la sección Replication configuration (Configuración de replicación), elija Edit (Editar).
5. Para eliminar todas las reglas de replicación, elija Delete all (Eliminar todo). Este paso requiere confirmación.

Para eliminar manualmente el rol vinculado a servicios mediante IAM

Utilice la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al AWSServiceRoleForECRReplicationservicio. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Rol vinculado a servicios de Amazon ECR para la caché de extracción

Amazon ECR utiliza un rol vinculado a un servicio denominado AWSServiceRoleForECRPullThroughCacheque permite a Amazon ECR realizar acciones en su nombre para completar las acciones de extracción de caché. Para obtener más información acerca de la caché de extracción, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.

Permisos de roles vinculados a servicios para Amazon ECR

El rol AWSServiceRoleForECRPullThroughCachevinculado al servicio confía en que el siguiente servicio asuma el rol.

- `pullthroughcache.ecr.amazonaws.com`

Detalles de los permisos

La política de permisos `AWSECRPullThroughCache_ServiceRolePolicy` se adjunta al rol vinculado al servicio. Esta política administrada otorga a Amazon ECR permiso para realizar las siguientes acciones. Para obtener más información, consulte `AWSECRPullThroughCache_ServiceRolePolicy`.

- `ecr`— Permite que el servicio Amazon ECR extraiga y envíe imágenes a un repositorio privado.

- `secretsmanager:GetSecretValue`— Permite que el servicio Amazon ECR recupere el contenido cifrado de un AWS Secrets Manager secreto. Esto es necesario cuando se utiliza una regla de caché de extracción para almacenar en caché las imágenes de un registro anterior que requiere autenticación en un registro privado. Este permiso solo se aplica a los secretos con el prefijo de nombre `ecr-pullthroughcache/`.

La política `AWSECRPullThroughCache_ServiceRolePolicy` contiene los siguientes JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECR",
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage",
                "ecr:BatchGetImage",
                "ecr:BatchImportUpstreamImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetImageCopyStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManager",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecr-pullthroughcache/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
```

```
}
```

Debe configurar permisos para permitir a una entidad de IAM (como, por ejemplo, un usuario, grupo o rol) crear, editar o eliminar la descripción de un rol vinculado a un servicio. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio para Amazon ECR

No es necesario crear manualmente el rol vinculado a servicios de Amazon ECR para la extracción de caché. Cuando crea una regla de extracción de caché para su registro privado en la AWS Management Console, la AWS CLI o la AWS API, Amazon ECR crea automáticamente la función vinculada al servicio.

Si elimina este rol vinculado a un servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Cuando crea una regla de caché de extracción para su registro privado, Amazon ECR crea el rol vinculado a servicios nuevamente para usted si aún no existe.

Edición de un rol vinculado a un servicio para Amazon ECR

Amazon ECR no permite editar manualmente el rol vinculado al AWSServiceRoleForECRPullThroughCacheservicio. Una vez creado el rol vinculado a servicios, no puede cambiar el nombre del rol porque varias entidades pueden hacer referencia a este. Sin embargo, puede editar la descripción del rol mediante IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio para Amazon ECR

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a servicios, recomendamos que elimine dicho rol. De esta forma no conservará una entidad no utilizada que no se monitorice ni se mantenga de forma activa. No obstante, debe eliminar las reglas de caché de extracción para su registro en cada región antes de poder eliminar manualmente el rol vinculado a servicios.

> **Note**
>
> Si intenta eliminar recursos mientras el servicio Amazon ECR sigue utilizando el rol, es posible que la acción de eliminación falle. Si eso sucede, espere unos minutos e inténtelo de nuevo.

Para eliminar los recursos de Amazon ECR utilizados por el rol vinculado a servicios AWSServiceRoleForECRPullThroughCache:

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región donde se crean las reglas de caché de extracción.
3. En el panel de navegación, elija Private registry (Registro privado).
4. En la página Private registry (Registro privado), en la sección Pull through cache configuration (Configuración de la caché de extracción), elija Edit (Editar).
5. Para cada regla de caché de extracción que haya creado, seleccione la regla y, a continuación, elija Delete rule (Eliminar regla).

Para eliminar manualmente el rol vinculado a servicios mediante IAM

Utilice la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al AWSServiceRoleForECRPullThroughCacheservicio. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Rol vinculado a servicios de Amazon ECR para plantillas de creación de repositorios

Amazon ECR utiliza un rol vinculado a un servicio denominado AWSServiceRoleForECRTemplateque permite a Amazon ECR realizar acciones en su nombre para completar las acciones de la plantilla de creación de repositorios.

Permisos de roles vinculados a servicios para Amazon ECR

El rol AWSServiceRoleForECRTemplatevinculado al servicio confía en que el siguiente servicio asuma el rol.

- `ecr.amazonaws.com`

Detalles de los permisos

La política de permisos `ECRTemplateServiceRolePolicy` se adjunta al rol vinculado al servicio. Esta política administrada da permiso a Amazon ECR para llevar a cabo acciones de creación de repositorios en su nombre.

La política `ECRTemplateServiceRolePolicy` contiene los siguientes JSON.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
     "Sid": "CreateRepositoryWithTemplate",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Debe configurar permisos para permitir a una entidad de IAM (como, por ejemplo, un usuario, grupo o rol) crear, editar o eliminar la descripción de un rol vinculado a un servicio. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio para Amazon ECR

No necesita crear manualmente el rol vinculado al servicio de Amazon ECR para plantillas de creación de repositorios. Cuando crea una regla de plantilla de creación de repositorios para su registro privado en la AWS Management Console AWS CLI, la o la AWS API, Amazon ECR crea automáticamente la función vinculada al servicio.

Si elimina este rol vinculado a un servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Cuando crea una regla de creación de repositorios para su registro privado, Amazon ECR crea el rol vinculado a servicios nuevamente para usted si aún no existe.

Edición de un rol vinculado a un servicio para Amazon ECR

Amazon ECR no permite editar manualmente el rol vinculado al AWSServiceRoleForECRTemplateservicio. Una vez creado el rol vinculado a servicios, no puede cambiar el nombre del rol porque varias entidades pueden hacer referencia a este. Sin embargo, puede editar la descripción del rol mediante IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio para Amazon ECR

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a servicios, recomendamos que elimine dicho rol. De esta forma no conservará una entidad no utilizada que no

se monitorice ni se mantenga de forma activa. No obstante, debe eliminar las reglas de creación de repositorios para su registro en cada región antes de poder eliminar manualmente el rol vinculado a servicios.

> **Note**
>
> Si intenta eliminar recursos mientras el servicio Amazon ECR sigue utilizando el rol, es posible que la acción de eliminación falle. Si eso sucede, espere unos minutos e inténtelo de nuevo.

Para eliminar los recursos de Amazon ECR utilizados por el rol vinculado a servicios AWSServiceRoleForECRTemplate:

1. Abra la consola Amazon ECR en https://console.aws.amazon.com/ecr/.
2. En la barra de navegación, elija la región en la que va a crear las reglas de creación de repositorios.
3. En el panel de navegación, elija Private registry (Registro privado).
4. En la página Registro privado, en la sección Planillas de creación de repositorios, elija Editar.
5. Para cada regla de creación de repositorios que haya creado, seleccione la regla y, a continuación, elija Eliminar regla.

Para eliminar manualmente el rol vinculado a servicios mediante IAM

Utilice la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al AWSServiceRoleForECRTemplateservicio. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Solución de problemas de identidad y acceso para Amazon Elastic Container Registry

Utilice la siguiente información para diagnosticar y solucionar los problemas habituales que es posible que surjan cuando se trabaja con Amazon ECR e IAM.

Temas

- No tengo autorización para realizar una acción en Amazon ECR

- No estoy autorizado a realizar lo siguiente: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Amazon ECR

No tengo autorización para realizar una acción en Amazon ECR

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `ecr:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `ecr:`*`GetWidget`*.

Si necesita ayuda, póngase en contacto con su administrador. AWS El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, sus políticas deben actualizarse a fin de permitirle pasar un rol a Amazon ECR.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Amazon ECR. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Amazon ECR

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admiten políticas basadas en recursos o listas de control de acceso (ACLs), puede utilizar esas políticas para permitir que las personas accedan a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon ECR admite estas características, consulte Cómo funciona Amazon Elastic Container Registry con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro de su propiedad en la Cuenta de AWS Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Protección de los datos en Amazon ECR

El modelo de se aplica a protección de datos en Amazon Elastic Container Service. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y

configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulta las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulta la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulta Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con Amazon ECS u otro Servicios de AWS dispositivo mediante la consola AWS CLI, la API o AWS SDKs. Cualquier dato que ingrese en etiquetas o campos de texto de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya la información de las credenciales en la URL para validar la solicitud para ese servidor.

Temas

- Cifrado en reposo

Cifrado en reposo

> **Important**
>
> El cifrado de doble capa del lado del servidor con AWS KMS (DSSE-KMS) solo está disponible en las regiones. AWS GovCloud (US)

Amazon ECR almacena imágenes en depósitos de Amazon S3 que administra Amazon ECR. De forma predeterminada, Amazon ECR utiliza el cifrado del lado del servidor con claves de cifrado administradas por Amazon S3 que cifran los datos en reposo mediante un algoritmo de cifrado AES-256. Esto no requiere ninguna acción por su parte y no conlleva ningún cargo adicional. Para obtener más información, consulte Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) (Protección de datos mediante el cifrado que se ejecuta en el servidor con las claves de cifrado administradas por Amazon S3 [SSE-S3]) en la Guía del usuario de Amazon Simple Storage Service.

Para tener más control sobre el cifrado de sus repositorios de Amazon ECR, puede utilizar el cifrado del lado del servidor con claves de KMS almacenadas en (). AWS Key Management Service AWS KMS Al cifrar AWS KMS los datos, puede utilizar la predeterminada, que gestiona Amazon ECR Clave administrada de AWS, o especificar su propia clave de KMS (denominada clave gestionada por el cliente). Para obtener más información, consulte Protección de datos mediante el cifrado del lado del servidor con claves de KMS almacenadas en AWS KMS (SSE-KMS) en la Guía del usuario de Amazon Simple Storage Service.

Puede optar por aplicar dos capas de cifrado a sus imágenes de Amazon ECR mediante el cifrado de doble capa del lado del servidor con (). AWS KMS DSSE-KMS DSSE-KMSLa opción es similar aSSE-KMS, pero aplica dos capas individuales de cifrado en lugar de una capa. Para obtener más información, consulte Uso del cifrado de doble capa del lado del servidor con AWS KMS claves (DSSE-KMS).

Cada repositorio de Amazon ECR tiene una configuración de cifrado, que se establece cuando este se crea. En cada repositorio se pueden utilizar diferentes configuraciones de cifrado. Para obtener más información, consulte Creación de un repositorio privado de Amazon ECR para almacenar imágenes.

Cuando se crea un repositorio con el AWS KMS cifrado activado, se utiliza una clave KMS para cifrar el contenido del repositorio. Además, Amazon ECR añade una AWS KMS concesión a la clave de KMS con el repositorio Amazon ECR como principal beneficiario.

A continuación se proporciona una descripción de alto nivel sobre cómo Amazon ECR se integra con AWS KMS para cifrar y descifrar sus repositorios:

1. Al crear un repositorio, Amazon ECR envía una DescribeKeyllamada AWS KMS a para validar y recuperar el nombre de recurso de Amazon (ARN) de la clave de KMS especificada en la configuración de cifrado.
2. Amazon ECR envía dos CreateGrantsolicitudes AWS KMS para crear concesiones en la clave de KMS para que Amazon ECR pueda cifrar y descifrar datos mediante la clave de datos.
3. Al insertar una imagen, se envía una GenerateDataKeysolicitud en la que se especifica la clave KMS AWS KMS que se utilizará para cifrar la capa de imágenes y el manifiesto.
4. AWS KMS genera una nueva clave de datos, la cifra con la clave KMS especificada y envía la clave de datos cifrada para que se almacene junto con los metadatos de la capa de imágenes y el manifiesto de la imagen.
5. Al extraer una imagen, se envía una solicitud de descifrado AWS KMS, especificando la clave de datos cifrados.
6. AWS KMS descifra la clave de datos cifrada y envía la clave de datos descifrados a Amazon S3.
7. La clave de datos se utiliza para descifrar la capa de imagen antes de que esta se extraiga.
8. Cuando se elimina un repositorio, Amazon ECR envía dos RetireGrantsolicitudes AWS KMS para retirar las concesiones creadas para el repositorio.

Consideraciones

Se deben tener en cuenta los siguientes puntos al utilizar el cifrado AWS KMS basado (SSE-KMSoDSSE-KMS) con Amazon ECR.

- Si crea su repositorio de Amazon ECR con cifrado de KMS y no especifica una clave de KMS, Amazon ECR utiliza una Clave administrada de AWS con el alias de forma `aws/ecr` predeterminada. Esta clave KMS se crea en su cuenta la primera vez que crea un repositorio con cifrado KMS habilitado.
- La configuración de cifrado del repositorio no se puede cambiar después de que se haya creado un repositorio.

- Cuando utiliza el cifrado KMS con su propia clave KMS, la clave debe existir en la misma región que el repositorio.
- No se deben revocar las concesiones que Amazon ECR crea en su nombre. Si revoca la autorización que autoriza a Amazon ECR a utilizar las AWS KMS claves de su cuenta, Amazon ECR no podrá acceder a estos datos, cifrar las nuevas imágenes introducidas en el repositorio ni descifrarlas cuando se extraigan. Al revocar una concesión para Amazon ECR, el cambio se produce inmediatamente. Para revocar derechos de acceso, debe eliminar el repositorio en lugar de revocar la concesión. Cuando se elimina un repositorio, Amazon ECR retira las concesiones en su nombre.
- El uso de las claves conlleva un coste. AWS KMS Para obtener más información, consulte Precios de AWS Key Management Service.
- Utilizar el cifrado del servidor de doble capa conlleva un costo asociado. Para obtener más información, consulte los precios de Amazon ECR

Permisos de IAM necesarios

Al crear o eliminar un repositorio de Amazon ECR con cifrado del lado del servidor mediante AWS KMS, los permisos necesarios dependen de la clave KMS específica utilice.

Permisos de IAM necesarios cuando se utiliza Clave administrada de AWS para Amazon ECR

De forma predeterminada, cuando el AWS KMS cifrado está habilitado para un repositorio de Amazon ECR pero no se especifica ninguna clave de KMS, se utiliza la Clave administrada de AWS de Amazon ECR. Cuando la clave AWS de KMS administrada para Amazon ECR se usa para cifrar un repositorio, cualquier responsable que tenga permiso para crear un repositorio también puede habilitar el AWS KMS cifrado en el repositorio. No obstante, la entidad de seguridad de IAM que elimina el repositorio debe tener el permiso `kms:RetireGrant`. Esto permite retirar las concesiones que se agregaron a la AWS KMS clave cuando se creó el repositorio.

El siguiente ejemplo de política de IAM se puede agregar como política insertada a un usuario para asegurarse de que tiene los permisos mínimos necesarios para eliminar un repositorio que tenga habilitado el cifrado. La clave KMS utilizada para cifrar el repositorio se puede especificar mediante el parámetro de recurso.

```
{
    "Version": "2012-10-17",
    "Id": "ecr-kms-permissions",
```

```
    "Statement": [
        {
            "Sid": "AllowAccessToRetireTheGrantsAssociatedWithTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:RetireGrant"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Permisos de IAM necesarios cuando se utiliza una clave administrada por el cliente

Al crear un repositorio con el AWS KMS cifrado habilitado mediante una clave administrada por el cliente, se requieren permisos tanto para la política de claves de KMS como para la política de IAM para el usuario o rol que crea el repositorio.

Al crear su propia clave KMS, puede usar la política de clave AWS KMS predeterminada o especificar la suya propia. Para garantizar que el propietario de la cuenta siga siendo administrable por el propietario de la cuenta, la política de claves de la clave KMS debe permitir todas AWS KMS las acciones del usuario raíz de la cuenta. Es posible que se agreguen permisos de ámbito adicionales a la política de clave, pero como mínimo se le deben dar permisos al usuario raíz para administrar la clave KMS. Para permitir que la clave KMS se use solo para las solicitudes que se originan en Amazon ECR, puede usar la clave de ViaService condición kms: con el `ecr.<region>.amazonaws.com` valor.

El siguiente ejemplo de política de claves otorga a la AWS cuenta (usuario raíz) propietaria de la clave de KMS acceso total a la clave de KMS. Para obtener más información sobre este ejemplo de política clave, consulte Permitir el acceso a la AWS cuenta y Habilitar las políticas de IAM en la Guía para AWS Key Management Service desarrolladores.

```
{
    "Version": "2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

El usuario de IAM, el rol de IAM o la AWS cuenta que crea los repositorios debe tener el `kms:DescribeKey` permiso y `kms:CreateGrantkms:RetireGrant`, además de los permisos de Amazon ECR necesarios.

> **Note**
>
> El permiso `kms:RetireGrant` se debe agregar a la política de IAM del usuario o rol que crea el repositorio. Los permisos `kms:CreateGrant` y `kms:DescribeKey` se pueden agregar a la política de clave para la clave KMS, o a la política de IAM del usuario o rol que crea el repositorio. Para obtener más información sobre cómo funcionan AWS KMS los permisos, consulte la referencia sobre permisos de AWS KMS API: acciones y recursos en la Guía para desarrolladores.AWS Key Management Service

El siguiente ejemplo de política de IAM se puede agregar como política insertada a un usuario, para asegurarse de que tiene los permisos mínimos necesarios para crear un repositorio que tenga habilitado el cifrado y borrarlo cuando haya terminado de utilizarlo. La AWS KMS key utilizada para cifrar el repositorio se puede especificar mediante el parámetro de recurso.

```
{
    "Version": "2012-10-17",
    "Id": "ecr-kms-permissions",
    "Statement": [
        {
            "Sid":
 "AllowAccessToCreateAndRetireTheGrantsAssociatedWithTheKeyAsWellAsDescribeTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:RetireGrant",
                "kms:DescribeKey"
            ],
```

```
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Habilitación de un usuario para mostrar una lista de claves KMS en la consola al crear un repositorio

Cuando utilice la consola de Amazon ECR para crear un repositorio, puede conceder permisos para permitir que un usuario publique las claves KMS administradas por el cliente en la región al habilitar el cifrado para el repositorio. En el siguiente ejemplo de política de IAM se muestran los permisos necesarios para mostrar una lista de las claves KMS y alias cuando se utiliza la consola.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
}
```

Monitoreo de la interacción de Amazon ECR con AWS KMS

Puede utilizarlo AWS CloudTrail para realizar un seguimiento de las solicitudes que Amazon ECR envía AWS KMS en su nombre. Las entradas de registro del CloudTrail registro contienen una clave de contexto de cifrado para que sean más fáciles de identificar.

Contexto de cifrado de Amazon ECR

Un contexto de cifrado es un conjunto de pares de clave-valor que contiene datos no secretos arbitrarios. Al incluir un contexto de cifrado en una solicitud de cifrado de datos, vincula AWS KMS criptográficamente el contexto de cifrado a los datos cifrados. Para descifrar los datos, es necesario pasar el mismo contexto de cifrado.

En sus solicitudes GenerateDataKeyy en las de Decrypt AWS KMS, Amazon ECR utiliza un contexto de cifrado con dos pares de nombre-valor que identifican el repositorio y el bucket de Amazon S3

que se utilizan. Esto se muestra en el siguiente ejemplo. Los nombres no varían, pero los valores de contexto de cifrado combinado serán diferentes para cada valor.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/
sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df",
    "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
}
```

Puede usar el contexto de cifrado para identificar estas operaciones criptográficas en registros y registros de auditoría, como AWS CloudTrailAmazon CloudWatch Logs, y como condición para la autorización en políticas y concesiones.

El contexto de cifrado de Amazon ECR se compone de dos pares de nombre-valor.

- aws:s3:arn: el primer par de nombre-valor identifica el bucket. La clave es `aws:s3:arn`. El valor es el Nombre de recurso de Amazon (ARN) del bucket de Amazon S3.

  ```
  "aws:s3:arn": "ARN of an Amazon S3 bucket"
  ```

  Por ejemplo, si el ARN del bucket fuera `arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`, el contexto de cifrado incluiría el siguiente par.

  ```
  "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
  ```

- aws:ecr:arn: el segundo par de nombre-valor identifica el Nombre de recurso de Amazon (ARN) del repositorio. La clave es `aws:ecr:arn`. El valor representa el ARN del proyecto.

  ```
  "aws:ecr:arn": "ARN of an Amazon ECR repository"
  ```

  Por ejemplo, si el ARN del repositorio fuera `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`, el contexto de cifrado incluiría el siguiente par.

  ```
  "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
  ```

Solución de problemas

Al eliminar un repositorio de Amazon ECR con la consola, si el repositorio se elimina correctamente pero Amazon ECR no puede retirar las concesiones agregadas a la clave KMS para su repositorio, recibirá el siguiente error.

```
The repository [{repository-name}] has been deleted successfully but the grants created
 by the kmsKey [{kms_key}] failed to be retired
```

Cuando esto ocurra, podrá retirar usted mismo las AWS KMS subvenciones para el repositorio.

Para retirar manualmente AWS KMS las subvenciones para un repositorio

1. Enumere las concesiones para la AWS KMS clave utilizada para el repositorio. El valor `key-id` se incluye en el error que recibe de la consola. También puede usar el `list-keys` comando para enumerar las claves de KMS Claves administradas por AWS y las administradas por el cliente en una región específica de su cuenta.

   ```
   aws kms list-grants \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
        --region us-west-2
   ```

   La salida incluye un elemento `EncryptionContextSubset` con el Nombre de recurso de Amazon (ARN) del repositorio. Esto se puede utilizar para determinar qué concesión agregada a la clave es la que desea retirar. En el siguiente paso se usará el valor `GrantId` al retirar la concesión.
2. Retira cada concesión de la AWS KMS clave agregada al repositorio. Sustituya el valor for por *`GrantId`* el ID de la concesión del resultado del paso anterior.

   ```
   aws kms retire-grant \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
        --grant-id GrantId \
        --region us-west-2
   ```

Validación de conformidad para Amazon Elastic Container Registry

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento

Servicios de AWS de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Cumplimiento de seguridad y gobernanza: en estas guías se explican las consideraciones de arquitectura y se proporcionan pasos para implementar las características de seguridad y cumplimiento.
- Referencia de servicios válidos de HIPAA: muestra una lista con los servicios válidos de HIPAA. No todos Servicios de AWS cumplen con los requisitos de la HIPAA.
- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.

- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Seguridad de la infraestructura en Amazon Elastic Container Registry

Como servicio gestionado, Amazon Elastic Container Registry está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a Amazon ECR a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puedes utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Puede llamar a estas operaciones de la API desde cualquier ubicación de red, pero Amazon ECR admite políticas de acceso basadas en recursos, que pueden incluir restricciones en función de la dirección IP de origen. También puede utilizar las políticas de Amazon ECR para controlar el acceso desde puntos de enlace específicos o específicos de Amazon Virtual Private Cloud (Amazon VPC). VPCs En efecto, esto aísla el acceso a la red a un recurso de Amazon ECR determinado únicamente de la VPC específica de la red. AWS Para obtener más información, consulte Puntos de enlace de VPC de la interfaz Amazon ECR ()AWS PrivateLink.

Puntos de enlace de VPC de la interfaz Amazon ECR ()AWS PrivateLink

Para mejorar la posición de seguridad de su VPC, configure Amazon ECR para que utilice un punto de enlace de la VPC de tipo interfaz. Los puntos de enlace de VPC funcionan con una tecnología que

le permite acceder de forma privada a Amazon ECR a APIs través de direcciones IP privadas. AWS PrivateLink AWS PrivateLink restringe todo el tráfico de red entre la VPC y Amazon ECR a la red de Amazon. No necesita una gateway de Internet, un dispositivo NAT ni una gateway privada virtual.

Para obtener más información sobre AWS PrivateLink los puntos de enlace de VPC, consulte los puntos de enlace de VPC en la Guía del usuario de Amazon VPC.

Consideraciones para los puntos de enlace de la VPC de Amazon ECR

Antes de configurar los puntos de enlace de la VPC para Amazon ECR, debe tener en cuenta las consideraciones siguientes.

- Para permitir que las tareas de Amazon ECS alojadas en EC2 instancias de Amazon extraigan imágenes privadas de Amazon ECR, cree los puntos de enlace de VPC de la interfaz para Amazon ECS. Para obtener más información, consulte Interface VPC Endpoints (AWS PrivateLink) en la Guía para desarrolladores de Amazon Elastic Container Service.
- Las tareas de Amazon ECS alojadas en Fargate que extraen imágenes de contenedor de Amazon ECR pueden restringir el acceso a la VPC específica que utilizan las tareas y al punto de enlace de la VPC que utiliza el servicio. Para ello, se agregan claves de condición al rol de IAM de ejecución de tareas para la tarea. Para obtener más información, consulte Permisos de IAM opcionales para tareas de Fargate que extraen imágenes de Amazon ECR sobre puntos de enlace de interfaz en la Guía para desarrolladores de Amazon Elastic Container Service.
- El grupo de seguridad asociado al punto de conexión de la VPC debe permitir las conexiones entrantes en el puerto 443 desde la subred privada de la VPC.
- Los puntos de enlace de la VPC no admiten las solicitudes entre regiones. Asegúrese de crear los puntos de enlace de la VPC en la misma región en la que tiene previsto enviar llamadas a la API de Amazon ECR.
- Los puntos de conexión de la VPC actualmente no admiten los repositorios públicos de Amazon ECR. Considere utilizar una regla de caché de extracción para alojar la imagen pública en un repositorio privado en la misma región que el punto de conexión de VPC. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.
- Los puntos de enlace de VPC solo admiten el DNS AWS proporcionado a través de Amazon Route 53. Si desea utilizar su propio DNS, puede utilizar el enrutamiento de DNS condicional. Para obtener más información, consulte Conjuntos de opciones de DHCP en la Guía del usuario de Amazon VPC.

- Si los contenedores tienen conexiones existentes a Amazon S3, sus conexiones podrían interrumpirse brevemente al agregar el punto de enlace de gateway de Amazon S3. Si desea evitar dicha interrupción, cree una nueva VPC que utilice el punto de enlace de gateway de Amazon S3 y luego migre el clúster de Amazon ECS y sus contenedores a la nueva VPC.
- Cuando se extrae una imagen mediante una regla de caché de extracción por primera vez, si ha configurado Amazon ECR para usar un punto de conexión de VPC de interfaz mediante AWS PrivateLink , entonces necesita crear una subred pública en la misma VPC, con una puerta de enlace NAT y, luego, dirigir todo el tráfico saliente a Internet desde su subred privada a la puerta de enlace NAT para que funcione la extracción. Las extracciones de imágenes posteriores no requieren esto. Para obtener más información, consulte Escenario: acceso a Internet desde una subred privada en la Guía del usuario de Amazon Virtual Private Cloud.

Consideraciones sobre las imágenes de Windows

Las imágenes basadas en el sistema operativo Windows incluyen artefactos cuya distribución está restringida por licencia. De forma predeterminada, al insertar imágenes de Windows a un repositorio de Amazon ECR, las capas que incluyen estos artefactos no se insertan, ya que se consideran capas externas. Cuando Microsoft proporciona los artefactos, las capas externas se recuperan de la infraestructura de Microsoft Azure. Por este motivo, para habilitar a los contenedores a de modo que puedan extraer estas capas externas de Azure, además de crear los puntos de enlace de la VPC se necesitarán completar más pasos.

Es posible anular este comportamiento al insertar imágenes de Windows a Amazon ECR mediante la marca `--allow-nondistributable-artifacts` en el daemon de Docker. Cuando se habilita, esta marca insertará las capas con licencia a Amazon ECR, lo que permite extraer estas imágenes de Amazon ECR a través del punto de enlace de la VPC sin que se necesite acceso adicional a Azure.

> **Important**
>
> El uso de la marca `--allow-nondistributable-artifacts` no excluye su obligación de cumplir con los términos de la licencia de imagen base de contenedor de Windows; no puede publicar contenido de Windows para redistribución pública o de terceros. Se permite el uso dentro de su propio entorno.

Para habilitar el uso de esta marca para la instalación de Docker, debe modificar el archivo de configuración del daemon de Docker que, en función de la instalación de Docker, normalmente se puede configurar en el menú de configuración o preferencias de la sección Docker Engine (Motor de Docker) o editando el archivo `C:\ProgramData\docker\config\daemon.json` directamente.

A continuación, mostramos un ejemplo de la configuración requerida. Reemplace el valor por el URI del repositorio al que está enviando imágenes.

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
    ]
}
```

Después de modificar el archivo de configuración del daemon de Docker, debe reiniciar este daemon antes de intentar insertar la imagen. Confirme que la inserción ha funcionado verificando que la capa base se ha integrado con su repositorio.

> **Note**
>
> Las capas base de las imágenes de Windows son grandes. El tamaño de la capa se traducirá en un mayor tiempo de inserción y en que estas imágenes conllevarán costos de almacenamiento adicionales en Amazon ECR. Por estas razones, se recomienda utilizar esta opción únicamente cuando sea estrictamente necesario reducir los tiempos de compilación y los costos de almacenamiento continuos. Por ejemplo, la imagen `mcr.microsoft.com/windows/servercore` tiene un tamaño aproximado de 1,7 GiB cuando se comprime en Amazon ECR.

Creación de puntos de enlace de la VPC para Amazon ECR

Para crear los puntos de enlace de la VPC para el servicio Amazon ECR, utilice el procedimiento Creación de un punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

Las tareas de Amazon ECS alojadas en las EC2 instancias de Amazon requieren tanto los puntos de enlace de Amazon ECR como el punto de enlace de la puerta de enlace de Amazon S3.

Las tareas de Amazon ECS alojadas en Fargate que utilizan la versión de la plataforma `1.4.0`, o una posterior, requieren los puntos de enlace de la VPC de Amazon ECR y los de gateway de Amazon S3.

Las tareas de Amazon ECS alojadas en Fargate que utilizan una versión de plataforma **`1.3.0`** o anterior solo requieren com.amazonaws. ***`region`***.ecr.dkr Punto de enlace de VPC Amazon ECR y puntos de enlace de puerta de enlace Amazon S3.

> **Note**
>
> El orden en el que se crean los puntos de enlace no importa.

com.amazonaws. ***`region`***.ecr.dkr

Este punto final se utiliza para el registro de Docker. APIs Los comandos del cliente de Docker como `push` y `pull` utilizan este punto de conexión.

Al crear este punto de enlace, debe habilitar un nombre de host de DNS privado. Para ello, asegúrese de que la opción Enable Private DNS Name (Habilitar nombre de DNS privado) está seleccionada en la consola de Amazon VPC al crear el punto de enlace de la VPC.

com.amazonaws. ***`region`***.ecr.api

> **Note**
>
> El especificado *`region`* representa el identificador de región de una AWS región compatible con Amazon ECR, como `us-east-2` la región EE.UU. Este (Ohio).

Este punto de enlace se utiliza para las llamadas a la API de Amazon ECR. Acciones de la API como `DescribeImages` y `CreateRepository` van a este punto de conexión.

Cuando se crea este punto de enlace, tiene la opción de habilitar un nombre de host DNS privado. Habilite esta configuración seleccionando Enable Private DNS Name (Habilitar nombre de DNS privado) en la consola de VPC al crear el punto de conexión de la VPC. Si habilitas un nombre de host DNS privado para el punto de enlace de la VPC, actualiza el SDK AWS CLI o a la versión más reciente para que no sea necesario especificar una URL de punto final cuando utilices el SDK AWS CLI .

Si habilitas un nombre de host DNS privado y utilizas un SDK o una AWS CLI versión publicada antes del 24 de enero de 2019, debes usar el `--endpoint-url` parámetro para especificar los puntos finales de la interfaz. En el ejemplo siguiente se muestra el formato de la dirección URL del punto de conexión.

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

Si no habilita un nombre de host de DNS privado para el punto de conexión de la VPC, debe utilizar el parámetro `--endpoint-url` que especifique el ID de punto de conexión de la VPC para el punto de conexión de interfaz. En el ejemplo siguiente se muestra el formato de la dirección URL del punto de conexión.

```
aws ecr create-repository --repository-name name --endpoint-url
 https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

Creación del punto de enlace de gateway de Amazon S3

Para que sus tareas de Amazon ECS extraigan imágenes privadas de Amazon ECR, debe crear un punto de enlace de gateway para Amazon S3. El punto de enlace de gateway es necesario porque Amazon ECR utiliza Amazon S3 para almacenar las capas de imágenes. Cuando sus contenedores descargan imágenes de Amazon ECR, deben acceder a Amazon ECR para obtener el manifiesto de imagen y a Amazon S3 para descargar las capas de imágenes en sí. A continuación, se muestra el Nombre de recurso de Amazon (ARN) del bucket de Amazon S3 que contiene las capas para cada imagen de Docker.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

Use el procedimiento Creación de un punto de enlace de gateway en la Guía del usuario de Amazon VPC para crear el siguiente punto de enlace de gateway de Amazon S3 para Amazon ECR. Cuando cree el punto de conexión, asegúrese de seleccionar las tablas de enrutamiento para su VPC.

com.amazonaws. ***region***.s3

El punto de enlace de gateway de Amazon S3 utiliza un documento de políticas de IAM para limitar el acceso al servicio. Se puede utilizar la política de Acceso completo, porque cualquier restricción que haya incluido en los roles de IAM de la tarea u otras políticas de usuario de IAM tienen vigencia sobre esta política. Si quiere limitar el acceso al bucket de Amazon S3 a los permisos mínimos necesarios para utilizar Amazon ECR, consulte Permisos mínimos del bucket de Amazon S3 para Amazon ECR.

Permisos mínimos del bucket de Amazon S3 para Amazon ECR

El punto de enlace de gateway de Amazon S3 utiliza un documento de políticas de IAM para limitar el acceso al servicio. Para permitir solo los permisos mínimos del bucket de Amazon S3 para Amazon ECR, restrinja el acceso al bucket de Amazon S3 que utiliza Amazon ECR al crear el documento de política de IAM para el punto de enlace.

En la siguiente tabla se describen los permisos de política de buckets de Amazon S3 necesarios para Amazon ECR.

Permiso	Descripción
`arn:aws:s3:::prod-` *`region`*`-starport-layer-bucket/*`	Proporciona acceso al bucket de Amazon S3 que contiene las capas para cada imagen de Docker. Representa el identificador de región de una región de AWS compatible con Amazon ECR, como lo es `us-east-2` para Este de EE. UU. (Ohio).

Ejemplo

El siguiente ejemplo ilustra cómo proporcionar acceso a los buckets de Amazon S3 necesarios para las operaciones de Amazon ECR.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

Cree el punto final de CloudWatch Logs

Las tareas de Amazon ECS que utilizan el tipo de lanzamiento Fargate que utilizan una VPC sin una puerta de enlace a Internet y que también utilizan el controlador de **awslogs** registro para enviar información de registro a CloudWatch Logs requieren que cree com.amazonaws. ***region***Punto final CloudWatch de VPC de la interfaz .logs para registros. Para obtener más información, consulte Uso de CloudWatch registros con puntos de enlace de VPC de interfaz en la Guía del usuario de Amazon CloudWatch Logs.

Creación de una política de punto de enlace para sus puntos de enlace de la VPC de Amazon ECR

Una política de punto de conexión de VPC es una política de recursos de IAM que puede asociar a un punto de conexión cuando crea o modifica el punto de conexión. Si no adjuntas una política al crear un punto de conexión, AWS adjunta una política predeterminada que te permita el acceso total al servicio. Una política de punto de conexión no anula ni sustituye a las políticas de usuario de ni las políticas específicas de los servicios. Se trata de una política independiente para controlar el acceso desde el punto de conexión al servicio especificado. Las políticas de punto de conexión deben escribirse en formato JSON. Para obtener más información, consulte Controlar el acceso a servicios con puntos de conexión de VPC en la Guía del usuario de Amazon VPC.

Recomendamos crear una única política de recursos de IAM y adjuntarla a ambos puntos de enlace de la VPC de Amazon ECR.

A continuación, se muestra un ejemplo de una política de punto de enlace para la Amazon ECR. Esta política permite a un rol de IAM específico extraer imágenes de Amazon ECR.

```
{
 "Statement": [{
  "Sid": "AllowPull",
  "Principal": {
   "AWS": "arn:aws:iam::1234567890:role/role_name"
  },
  "Action": [
   "ecr:BatchGetImage",
   "ecr:GetDownloadUrlForLayer",
                  "ecr:GetAuthorizationToken"
  ],
  "Effect": "Allow",
  "Resource": "*"
```

```
 }]
}
```

El siguiente ejemplo de política de punto de conexión impide que se elimine un repositorio especificado.

```
{
 "Statement": [{
   "Sid": "AllowAll",
   "Principal": "*",
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*"
  },
  {
   "Sid": "PreventDelete",
   "Principal": "*",
   "Action": "ecr:DeleteRepository",
   "Effect": "Deny",
   "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  }
 ]
}
```

El siguiente ejemplo de política de punto de conexión combina los dos ejemplos anteriores en una única política.

```
{
 "Statement": [{
   "Sid": "AllowAll",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "*",
   "Resource": "*"
  },
  {
   "Sid": "PreventDelete",
   "Effect": "Deny",
   "Principal": "*",
   "Action": "ecr:DeleteRepository",
   "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  },
```

```
  {
   "Sid": "AllowPull",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::1234567890:role/role_name"
   },
   "Action": [
    "ecr:BatchGetImage",
    "ecr:GetDownloadUrlForLayer",
                          "ecr:GetAuthorizationToken"
   ],
   "Resource": "*"
  }
 ]
}
```

Modificación de la política de punto de enlace de la VPC para Amazon ECR

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, elija Puntos de conexión.
3. Si aún no ha creado los puntos de enlace de la VPC para Amazon ECR, consulte Creación de puntos de enlace de la VPC para Amazon ECR.
4. Seleccione el punto de enlace de la VPC de Amazon ECR al que desea agregar una política y elija la pestaña Policy (Política) en la mitad inferior de la pantalla.
5. Elija Editar política y realice los cambios en la política.
6. Elija Guardar para guardar el cambio.

Subredes compartidas

No puede crear, describir, modificar ni eliminar puntos de conexión de VPC en subredes que se compartan con usted. No obstante, puede usar los puntos de conexión de VPC en las subredes que se compartan con usted.

Prevención de la sustitución confusa entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede provocar un confuso problema

de diputado. La suplantación entre servicios puedes producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puedes manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Se recomienda utilizar las claves de contexto de condición global `aws:SourceArn` o `aws:SourceAccount` en las políticas de recursos para limitar los permisos que Amazon ECR concede a otro servicio para el recurso. Utiliza `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utiliza `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*`.

Si el valor de `aws:SourceArn` no contiene el ID de cuenta, como un ARN de bucket de Amazon S3, debe utilizar ambas claves de contexto de condición global para limitar los permisos.

El valor `aws:SourceArn` debe ser ResourceDescription.

El siguiente ejemplo muestra cómo puede utilizar las claves de contexto de condición `aws:SourceAccount` global `aws:SourceArn` y las claves de contexto de una política de repositorio de Amazon ECR para permitir el AWS CodeBuild acceso a las acciones de la API de Amazon ECR necesarias para la integración con ese servicio y, al mismo tiempo, evitar el confuso problema de los diputados.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
```

```
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:region:123456789012:project/project-
name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

Monitoreo de Amazon ECR

Puede supervisar el uso de la API de Amazon ECR con Amazon CloudWatch, que recopila y procesa datos sin procesar de Amazon ECR para convertirlos en métricas legibles y prácticamente en tiempo real. Estas estadísticas se registran durante un periodo de dos semanas para que pueda acceder a información histórica y obtener una perspectiva sobre el uso de la API. Los datos métricos de Amazon ECR se envían automáticamente CloudWatch en períodos de un minuto. Para obtener más información al respecto CloudWatch, consulta la Guía del CloudWatch usuario de Amazon.

Amazon ECR proporciona métricas basadas en el uso de la API para acciones de autorización, inserción de imágenes y extracción de imágenes.

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon ECR y sus AWS soluciones. Le recomendamos que recopile datos de monitoreo de los recursos que componen su AWS solución para poder depurar más fácilmente un error multipunto en caso de que se produzca. No obstante, antes de comenzar a monitorear Amazon ECR, debe crear un plan que incluya respuestas a las siguientes preguntas:

- ¿Cuáles son los objetivos de la supervisión?
- ¿Qué recursos va a supervisar?
- ¿Con qué frecuencia va a supervisar estos recursos?
- ¿Qué herramientas de supervisión va a utilizar?
- ¿Quién se encargará de realizar las tareas de supervisión?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso consiste en establecer un punto de referencia del rendimiento normal de Amazon ECR en su entorno. Para ello, se debe medir el rendimiento en distintos momentos y bajo distintas condiciones de carga. A medida que monitoree Amazon ECR, guarde los datos de monitoreo históricos para que pueda compararlos con los datos de rendimiento actual, identificar los patrones de rendimiento normal y las anomalías en el rendimiento, así como desarrollar métodos para la resolución de problemas.

Temas

- Visualización de Service Quotas y configuración de alarmas
- Métricas de uso de Amazon ECR

- Informes de uso de Amazon ECR
- Métricas de repositorios de Amazon ECR
- Eventos de Amazon ECR y EventBridge
- Registrar las acciones de Amazon ECR con AWS CloudTrail

Visualización de Service Quotas y configuración de alarmas

Puedes usar la CloudWatch consola para visualizar tus cuotas de servicio y comparar tu uso actual con las cuotas de servicio. También puede configurar alarmas para recibir notificaciones cuando se acerque a una cuota.

Para visualizar una cuota de servicio y, opcionalmente, configurar una alarma

1. Abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, seleccione Métricas.
3. En la pestaña All metrics (Todas las métricas), elija Usage (Uso) y, a continuación, By AWS Resource (Por recurso de AWS).

   Aparecerá la lista de métricas de uso de cuotas de servicio.
4. Active la casilla situada junto a una de las métricas.

   El gráfico muestra el uso actual de ese AWS recurso.
5. Para añadir su cuota de servicio al gráfico, haga lo siguiente:

   a. Elija la pestaña Métricas diagramadas.

   b. Elija Math expression (Expresión matemática) y Start with an empty expression (Comenzar con una expresión vacía). A continuación, en la nueva fila, en Details (Detalles), escriba **`SERVICE_QUOTA(m1)`**.

      Se añade una nueva línea al gráfico, mostrando la cuota de servicio del recurso representado en la métrica.
6. Para ver su uso actual como porcentaje de la cuota, añada una nueva expresión o cambie la expresión SERVICE_QUOTA actual. En el caso de la nueva expresión, use **`m1/60/SERVICE_QUOTA(m1)*100`**
7. (Opcional) Para configurar una alarma que le notifique si se acerca a la cuota de servicio, haga lo siguiente:

a. En la fila **`m1/60/SERVICE_QUOTA(m1)*100`**, en Actions (Acciones), elija el icono de alarma. Se parece a una campana.

   Aparecerá la página de creación de alarmas.

b. En Conditions (Condiciones), asegúrese de que Threshold type (Tipo de umbral) es Static (Estático) y Whenever Expression1 is (Siempre que Expression1 sea) se establece en Greater (Mayor). En than (que), escriba **`80`**. Esto crea una alarma que pasa al estado ALARM cuando su uso supera el 80 % de la cuota.

c. Elija Siguiente.

d. En la página siguiente, seleccione un tema de Amazon SNS o cree uno. Este tema se notifica cuando la alarma pasa al estado ALARM. A continuación, elija Siguiente.

e. En la página siguiente, escriba un nombre y una descripción para la alarma y, a continuación, elija Next (Siguiente).

f. Elija Crear alarma.

Métricas de uso de Amazon ECR

Puedes usar las métricas CloudWatch de uso para proporcionar visibilidad sobre el uso de los recursos de tu cuenta. Usa estas métricas para visualizar tu uso actual del servicio en CloudWatch gráficos y paneles.

Las métricas de uso de Amazon ECR corresponden a las cuotas AWS de servicio. Puede configurar alarmas que le avisen cuando su uso se acerque a una Service Quota. Para obtener más información acerca de las cuotas de servicio de Amazon ECR, consulte Cuotas de servicio de Amazon ECR.

Amazon ECR publica las siguientes métricas en el espacio de nombres `AWS/Usage`.

Métrica	Descripción
`CallCount`	Número de llamadas de acciones de la API realizadas desde la cuenta. Los recursos se definen por las dimensiones asociadas a la métrica.

Métrica	Descripción
	La estadística más útil de esta métrica es `SUM`, que represent a la suma de los valores de todas las contribuciones durante el período definido.

Las siguientes dimensiones se utilizan para ajustar las métricas de uso que publica Amazon ECR.

Dimensión	Descripción
`Service`	El nombre del AWS servicio que contiene el recurso. En el caso de las métricas de uso de Amazon ECR, el valor de esta dimensión es `ECR`.
`Type`	El tipo de entidad que se registra. Actualmente, el único valor válido para las métricas de uso de Amazon ECR es `API`.
`Resource`	El tipo de recurso que se está ejecutando. Actualmente, Amazon ECR devuelve información sobre el uso de la API en relación con las siguientes acciones. • `GetAuthorizationToken` • `BatchCheckLayerAvailability` • `InitiateLayerUpload` • `UploadLayerPart` • `CompleteLayerUpload` • `PutImage` • `BatchGetImage` • `GetDownloadUrlForLayer`
`Class`	La clase de recurso a la que se realiza el seguimiento. En la actualidad, Amazon ECR no utiliza la dimensión de clase.

Informes de uso de Amazon ECR

AWS proporciona una herramienta de informes gratuita llamada Cost Explorer que le permite analizar el costo y el uso de los recursos de Amazon ECR.

Utilice Cost Explorer para ver gráficos sobre el uso y los costos. Puede ver los datos de los 13 meses anteriores y predecir la cantidad que probablemente va a gastar durante los tres meses siguientes. Puede utilizar el Cost Explorer para ver sus patrones de gasto en recursos de AWS a lo largo del tiempo, identificar aspectos que deben estudiarse más a fondo y consultar tendencias que le pueden ayudar a comprender los costos. También puede especificar intervalos de tiempo en los datos y ver los datos temporales por día o por mes.

Los datos de medición de los informes de uso y costes muestran el uso en todos los repositorios de Amazon ECR. Para obtener más información, consulte Etiquetado de los recursos para facturación.

Para obtener más información sobre la creación de un informe de AWS costos y uso, consulte el informe de AWS costos y uso en la Guía del AWS Billing usuario.

Métricas de repositorios de Amazon ECR

Amazon ECR envía las métricas del recuento de extracciones del repositorio a Amazon CloudWatch. Los datos métricos de Amazon ECR se envían automáticamente CloudWatch en períodos de 1 minuto. Para obtener más información al respecto CloudWatch, consulta la Guía del CloudWatch usuario de Amazon.

Temas

- Habilitar CloudWatch las métricas
- Métricas y dimensiones disponibles
- Visualización de las métricas de Amazon ECR mediante la consola CloudWatch

Habilitar CloudWatch las métricas

Amazon ECR envía métricas de repositorio automáticamente para todos los repositorios. No es necesario tomar medidas de forma manual.

Métricas y dimensiones disponibles

En las siguientes secciones se enumeran las métricas y dimensiones que Amazon ECR envía a Amazon CloudWatch.

Métricas de Amazon ECR

Amazon ECR proporciona métricas para que usted pueda monitorear sus repositorios. Puede medir el recuento de extracciones.

El espacio de nombres de `AWS/ECR` incluye las siguientes métricas.

`RepositoryPullCount`

El número total de extracciones de las imágenes en el repositorio.

Dimensiones válidas: `RepositoryName`.

Estadísticas válidas: Average, Minimum, Maximum, Sum, Sample Count. La estadística más útil es Suma.

Unidad: entero.

Dimensiones para las métricas de Amazon ECR

Las métricas de Amazon ECR utilizan el espacio de nombres `AWS/ECR` y proporcionan métricas para las dimensiones siguientes.

`RepositoryName`

Esta dimensión filtra los datos solicitados de todas las imágenes del contenedor en un repositorio específico.

Visualización de las métricas de Amazon ECR mediante la consola CloudWatch

Puede ver las métricas del repositorio de Amazon ECR en la CloudWatch consola. La CloudWatch consola proporciona una visualización detallada y personalizable de sus recursos. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

Eventos de Amazon ECR y EventBridge

Amazon EventBridge le permite automatizar sus AWS servicios y responder automáticamente a los eventos del sistema, como los problemas de disponibilidad de las aplicaciones o los cambios de recursos. Los eventos de AWS los servicios se entregan EventBridge prácticamente en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés e incluir las acciones automatizadas que deben realizarse cuando un evento cumpla una de las reglas. Entre las acciones que se pueden activar automáticamente se incluyen las siguientes:

- Añadir eventos a grupos de CloudWatch registros en Logs
- Invocar una función AWS Lambda
- Invocar el comando Amazon EC2 Run
- Desviar el evento a Amazon Kinesis Data Streams
- Activar una máquina de AWS Step Functions estados
- Notificar un tema de Amazon SNS o una cola de Amazon SQS

Para obtener más información, consulta Cómo empezar con Amazon EventBridge en la Guía del EventBridge usuario de Amazon.

Eventos de ejemplo de Amazon ECR

Los siguientes ejemplos corresponden a eventos de Amazon ECR. Los eventos se emiten en la medida de lo posible.

Evento para una inserción de imagen completada

El siguiente evento se envía cuando se completa cada inserción de imagen. Para obtener más información, consulte Inserción de una imagen de Docker en un repositorio privado de Amazon ECR.

```
{
    "version": "0",
    "id": "13cde686-328b-6117-af20-0e5566167482",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-11-16T01:54:34Z",
    "region": "us-west-2",
    "resources": [],
```

```
    "detail": {
        "result": "SUCCESS",
        "repository-name": "my-repository-name",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "PUSH",
        "image-tag": "latest"
    }
}
```

Evento para una acción de caché de extracción

El siguiente evento se envía cuando se intenta realizar una acción de caché de extracción. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.

```
{
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Pull Through Cache Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2023-02-29T02:36:48Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecr:us-west-2:123456789012:repository/docker-hub/alpine"
    ],
    "detail": {
        "rule-version": "1",
        "sync-status": "SUCCESS",
        "ecr-repository-prefix": "docker-hub",
        "repository-name": "docker-hub/alpine",
        "upstream-registry-url": "public.ecr.aws",
        "image-tag": "3.17.2",
        "image-digest":
 "sha256:4aa08ef415aecc80814cb42fa41b658480779d80c77ab15EXAMPLE",
    }
}
```

Evento para un escaneo de imágenes completado (escaneo básico)

Cuando el escaneo básico está habilitado para su registro, se envía el siguiente evento cuando se completa cada escaneo de imágenes. El parámetro `finding-severity-counts` solo devolverá

un valor para un nivel de gravedad, si existe. Por ejemplo, si la imagen no contiene resultados de nivel `CRITICAL`, no se devolverá ningún recuento crítico. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades de sistema operativo en Amazon ECR.

> **Note**
>
> Para obtener más información sobre los eventos que Amazon Inspector emite cuando se habilita el escaneo mejorado, consulte EventBridge eventos enviados para un escaneo mejorado en Amazon ECR.

```
{
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Image Scan",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-10-29T02:36:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecr:us-east-1:123456789012:repository/my-repository-name"
    ],
    "detail": {
        "scan-status": "COMPLETE",
        "repository-name": "my-repository-name",
        "finding-severity-counts": {
        "CRITICAL": 10,
        "MEDIUM": 9
      },
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "image-tags": []
    }
}
```

Evento de notificación de cambio en un recurso con escaneo mejorado habilitado (escaneo mejorado)

Cuando el escaneo mejorado está habilitado para su registro, Amazon ECR envía el siguiente evento cuando se produce un cambio con un recurso que tiene habilitado el escaneo mejorado. Esto incluye la creación de nuevos repositorios, la frecuencia de escaneo de un repositorio que

se está modificando o cuando se crean o eliminan imágenes en repositorios con el escaneo mejorado habilitado. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0ab8638973bf",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
   {
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

Evento para una eliminación de imagen

El siguiente evento se envía cuando se elimina una imagen. Para obtener más información, consulte Eliminación de una imagen en Amazon ECR.

```
{
    "version": "0",
    "id": "dd3b46cb-2c74-f49e-393b-28286b67279d",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-11-16T02:01:05Z",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "result": "SUCCESS",
        "repository-name": "my-repository-name",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "DELETE",
        "image-tag": "latest"
    }
}
```

Evento para una replicación de una imagen completada

El siguiente evento se envía cuando se completa una replicación de imagen. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR.

```
{
  "version": "0",
  "id": "c8b133b1-6029-ee73-e2a1-4f466b8ba999",
  "detail-type": "ECR Replication Action",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2024-05-08T20:44:54Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecr:us-east-1:123456789012:repository/docker-hub/alpine"
  ],
  "detail": {
    "result": "SUCCESS",
    "repository-name": "docker-hub/alpine",
    "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
    "source-account": "123456789012",
    "action-type": "REPLICATE",
```

```
    "source-region": "us-west-2",
    "image-tag": "3.17.2"
  }
}
```

Registrar las acciones de Amazon ECR con AWS CloudTrail

Amazon ECR está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Amazon ECR. CloudTrail captura las siguientes acciones de Amazon ECR como eventos:

- Todas las llamadas a la API, incluidas las llamadas desde la consola de Amazon ECR
- Todas las acciones realizadas debido a la configuración de cifrado en sus repositorios
- Todas las acciones realizadas debido a las reglas de la directiva del ciclo de vida, incluidas las acciones correctas y las que no tienen éxito

> **Important**
>
> Debido a las limitaciones de tamaño de los CloudTrail eventos individuales, para las acciones de política de ciclo de vida en las que 10 o más imágenes han caducado, Amazon ECR envía varios eventos a CloudTrail. Además, Amazon ECR incluye un máximo de 100 etiquetas por imagen.

Cuando se crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Amazon ECR. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con esta información, se puede determinar la solicitud que se envió a Amazon ECR, la dirección IP de origen, quién realizó la solicitud, cuándo la realizó, así como detalles adicionales.

Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.

Información de Amazon ECR en CloudTrail

CloudTrail está habilitada en su AWS cuenta al crear la cuenta. Cuando se produce una actividad en Amazon ECR, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su

AWS cuenta. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos de su AWS cuenta, incluidos los eventos de Amazon ECR, cree un registro. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. Al crear un registro de seguimiento en la consola, puede aplicarlo a una sola región o a todas las regiones. La ruta registra los eventos en la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para obtener más información, consulte:

- Crear una ruta para tu AWS cuenta
- AWS integraciones de servicios con registros CloudTrail
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas las acciones de la API ECR de Amazon se registran CloudTrail y se documentan en la referencia de la API de Amazon Elastic Container Registry. Al realizar tareas comunes, se generan secciones en los archivos de CloudTrail registro para cada acción de la API que forma parte de esa tarea. Por ejemplo, cuando se crea un repositorio `CreateRepository` y `GetAuthorizationToken` se generan `SetRepositoryPolicy` secciones en los archivos de CloudTrail registro. Cuando inserta una imagen en un repositorio, se generan las secciones `InitiateLayerUpload`, `UploadLayerPart`, `CompleteLayerUpload` y `PutImage`. Cuando extrae una imagen, se generan las secciones `GetDownloadUrlForLayer` y `BatchGetImage`. Cuando OCI los clientes que admiten la OCI 1.1 especificación obtienen la lista de referencias (o artefactos de referencia) de una imagen mediante la API de referencias, se emite un evento. `ListImageReferrers` CloudTrail Si desea ver ejemplos de estas tareas comunes, consulte CloudTrail ejemplos de entradas de registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- si la solicitud se realizó con las credenciales del nodo raíz o del usuario
- si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado
- Si la solicitud la realizó otro servicio AWS

Para obtener más información, consulte el elemento `userIdentity` de CloudTrail.

Descripción de las entradas de archivos de registros de Amazon ECR

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud y otra información. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

CloudTrail ejemplos de entradas de registro

Los siguientes son ejemplos de entradas de CloudTrail registro para algunas tareas comunes de Amazon ECR.

Estos ejemplos se han manipulado para mejorar la legibilidad. En un archivo de CloudTrail registro, todas las entradas y eventos se concatenan en una sola línea. Además, este ejemplo se ha limitado a una única entrada de Amazon ECR. En un archivo de CloudTrail registro real, puede ver las entradas y los eventos de varios servicios. AWS

> **Important**
>
> La fuente IPAddress es la dirección IP desde la que se realizó la solicitud. Para las acciones que se originan desde la consola del servicio, la dirección registrada es para su recurso subyacente, no para el servidor web de la consola. En el caso de los servicios en AWS, solo se muestra el nombre DNS. Aun así, evaluamos la autenticación con la IP de origen del cliente, incluso si está redactada con el nombre de DNS del servicio de AWS .

Temas

- Ejemplo: Acción de creación de repositorio
- Ejemplo: acción AWS KMSCreateGrant de la API al crear un repositorio de Amazon ECR
- Ejemplo: Acción de inserción de imágenes
- Ejemplo: Acción de extracción de imágenes
- Ejemplo: Acción de política de ciclo de vida de imágenes
- Ejemplo: acción de referencia de imágenes

Ejemplo: Acción de creación de repositorio

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `CreateRepository` acción.


```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-11T21:54:07Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            }
        }
    },
    "eventTime": "2018-07-11T22:17:43Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "CreateRepository",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "repositoryName": "testrepo"
    },
    "responseElements": {
        "repository": {
            "repositoryArn": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "repositoryName": "testrepo",
            "repositoryUri": "123456789012.dkr.ecr.us-east-2.amazonaws.com/testrepo",
            "createdAt": "Jul 11, 2018 10:17:44 PM",
            "registryId": "123456789012"
        }
```

```
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "accountId": "123456789012"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Ejemplo: acción AWS KMS**`CreateGrant`** de la API al crear un repositorio de Amazon ECR

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la AWS KMS `CreateGrant` acción que se debe realizar al crear un repositorio de Amazon ECR con el cifrado KMS activado. Por cada repositorio que se cree con el cifrado KMS activado, debería ver dos entradas de `CreateGrant` registro. CloudTrail

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIEP6W46J43IG7LXAQ",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {

            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-06-10T19:22:10Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
```

```
    "eventTime": "2020-06-10T19:22:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "keyId": "4b55e5bf-39c8-41ad-b589-18464af7758a",
        "granteePrincipal": "ecr.us-west-2.amazonaws.com",
        "operations": [
            "GenerateDataKey",
            "Decrypt"
        ],
        "retiringPrincipal": "ecr.us-west-2.amazonaws.com",
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecr:arn": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo"
            }
        }
    },
    "responseElements": {
        "grantId": "3636af9adfee1accb67b83941087dcd45e7fadc4e74ff0103bb338422b5055f3"
    },
    "requestID": "047b7dea-b56b-4013-87e9-a089f0f6602b",
    "eventID": "af4c9573-c56a-4886-baca-a77526544469",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:123456789012:key/4b55e5bf-39c8-41ad-
b589-18464af7758a"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Ejemplo: Acción de inserción de imágenes

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que muestra una inserción de imágenes en la que se utiliza la `PutImage` acción.

> **Note**
>
> Al insertar una imagen, también verás `InitiateLayerUploadUploadLayerPart`, y `CompleteLayerUpload` referencias en los CloudTrail registros.


```
{
    "eventVersion": "1.04",
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "PutImage",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageTag": "latest",
  "registryId": "123456789012",
  "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
```

```
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest\":
 \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n      },
\n      {\n         \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",\n         \"size\": 37720774,\n         \"digest\":
 \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
   ]\n}"
 },
 "responseElements": {
  "image": {
   "repositoryName": "testrepo",
   "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
```

```
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest\":
 \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n      },
\n      {\n         \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",\n         \"size\": 37720774,\n         \"digest\":
 \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
   ]\n}",
   "registryId": "123456789012",
   "imageId": {
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
    "imageTag": "latest"
   }
```

```
  }
 },
 "requestID": "cf044b7d-5f9d-11e9-9b2a-95983139cc57",
 "eventID": "2bfd4ee2-2178-4a82-a27d-b12939923f0f",
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
}
```

Ejemplo: Acción de extracción de imágenes

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro en la que se muestra una extracción de imágenes que utiliza la `BatchGetImage` acción.

> **Note**
>
> Cuando extraiga una imagen, si aún no la tiene localmente, verá también referencias `GetDownloadUrlForLayer` en los registros de CloudTrail .

```
{
    "eventVersion": "1.04",
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T17:23:20Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "BatchGetImage",
```

```
 "awsRegion": "us-east-2",
 "sourceIPAddress": "ecr.amazonaws.com",
 "userAgent": "ecr.amazonaws.com",
 "requestParameters": {
  "imageIds": [{
   "imageTag": "latest"
  }],
  "acceptedMediaTypes": [
   "application/json",
   "application/vnd.oci.image.manifest.v1+json",
   "application/vnd.oci.image.index.v1+json",
   "application/vnd.docker.distribution.manifest.v2+json",
   "application/vnd.docker.distribution.manifest.list.v2+json",
   "application/vnd.docker.distribution.manifest.v1+prettyjws"
  ],
  "repositoryName": "testrepo",
  "registryId": "123456789012"
 },
 "responseElements": null,
 "requestID": "2a1b97ee-5fa3-11e9-a8cd-cd2391aeda93",
 "eventID": "c84f5880-c2f9-4585-9757-28fa5c1065df",
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
}
```


Ejemplo: Acción de política de ciclo de vida de imágenes

El siguiente ejemplo muestra una entrada de CloudTrail registro que muestra cuándo ha caducado una imagen debido a una regla de política de ciclo de vida. Este tipo de evento se puede encontrar filtrando por `PolicyExecutionEvent` en el campo de nombre del evento.

Al probar una vista previa de una política de ciclo de vida, Amazon ECR genera una entrada de CloudTrail registro con el campo del nombre del evento de`DryRunEvent`, con exactamente la misma estructura que la`PolicyExecutionEvent`. Si cambia el nombre del evento a`DryRunEvent`, puede filtrar los eventos de prueba en su lugar.

> **Important**
>
> Debido a las limitaciones de tamaño de los CloudTrail eventos individuales, para las acciones de política de ciclo de vida en las que 10 o más imágenes han caducado, Amazon ECR envía varios eventos a CloudTrail. Además, Amazon ECR incluye un máximo de 100 etiquetas por imagen.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-03-12T20:22:12Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "PolicyExecutionEvent",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "9354dd7f-9aac-4e9d-956d-12561a4923aa",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
            "accountId": "123456789012",
            "type": "AWS::ECR::Repository"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "repositoryName": "testrepo",
        "lifecycleEventPolicy": {
            "lifecycleEventRules": [
                {
                    "rulePriority": 1,
                    "description": "remove all images > 2",
                    "lifecycleEventSelection": {
                        "tagStatus": "Any",
```

```
                        "tagPrefixList": [],
                        "countType": "Image count more than",
                        "countNumber": 2
                    },
                    "action": "expire"
                }
            ],
            "lastEvaluatedAt": 0,
            "policyVersion": 1,
            "policyId": "ceb86829-58e7-9498-920c-aa042e33037b"
        },
        "lifecycleEventImageActions": [
            {
                "lifecycleEventImage": {
                    "digest":
"sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45",
                    "tagStatus": "Tagged",
                    "tagList": [
                        "alpine"
                    ],
                    "pushedAt": 1584042813000
                },
                "rulePriority": 1
            },
            {
                "lifecycleEventImage": {
                    "digest":
"sha256:6ab380c5a5acf71c1b6660d645d2cd79cc8ce91b38e0352cbf9561e050427baf",
                    "tagStatus": "Tagged",
                    "tagList": [
                        "centos"
                    ],
                    "pushedAt": 1584042842000
                },
                "rulePriority": 1
            }
        ],
        "lifecycleEventFailureDetails": [
            {
                "lifecycleEventImage": {
                    "digest":
"sha256:9117e1bc28cd20751e584b4ccd19b1178d14cf02d134b04ce6be0cc51bff762a",
                    "tagStatus": "Untagged",
                    "tagList": [],
```

```
                    "pushedAt": 1584042844000
                },
                "rulePriority": 1,
                "failureCode": "ImageReferencedByManifestList",
                "failureReason": "Requested image referenced by manifest list:
 [sha256:4b27c83d44a18c31543039d9e8b2786043ec6c8d00804d5800c5148d6b6f65bc]"
            }
        ]
    }
}
```

Ejemplo: acción de referencia de imágenes

En el siguiente ejemplo, se muestra una entrada de AWS CloudTrail registro que demuestra cuándo un cliente que OCI 1.1 cumple con las normas busca una lista de referencias (o artefactos de referencia) de una imagen mediante la API. `Referrers`

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-10-08T16:38:39Z",
                "mfaAuthenticated": "false"
            },
            "ec2RoleDelivery": "2.0"
        },
        "invokedBy": "ecr.amazonaws.com"
    },
    "eventTime": "2024-10-08T17:22:51Z",
```

```
    "eventSource": "ecr.amazonaws.com",
    "eventName": "ListImageReferrers",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "ecr.amazonaws.com",
    "userAgent": "ecr.amazonaws.com",
    "requestParameters": {
        "registryId": "123456789012",
        "repositoryName": "testrepo",
        "subjectId": {
            "imageDigest":
 "sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a"
        },
        "nextToken": "urD72mdD/mC8b5-EXAMPLE"
    },
    "responseElements": null,
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123456789012",
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Uso de Amazon ECR con un SDK AWS

AWS Los kits de desarrollo de software (SDKs) están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ ejemplos de código
AWS CLI	AWS CLI ejemplos de código
AWS SDK para Go	AWS SDK para Go ejemplos de código
AWS SDK para Java	AWS SDK para Java ejemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK para .NET	AWS SDK para .NET ejemplos de código
AWS SDK para PHP	AWS SDK para PHP ejemplos de código
Herramientas de AWS para PowerShell	Herramientas de AWS para PowerShell ejemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) ejemplos de código
AWS SDK para Ruby	AWS SDK para Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

Ejemplos de código para Amazon ECR mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar Amazon ECR con un kit de desarrollo de AWS software (SDK).

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon ECR con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción

Introducción a Amazon ECR

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;
```

```
public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Para obtener detalles sobre la API, consulte listImages en la Referencia de la API de AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
```

```
        }
    }
}
```

- Para obtener información sobre la API, consulte listImages en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
```

```
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Para obtener más información sobre la API, consulta ListImages en la referencia de la API AWS del SDK for Python (Boto3).

Ejemplos de código

- Ejemplos básicos de Amazon ECR mediante AWS SDKs
  - Introducción a Amazon ECR
  - Conozca los conceptos básicos de Amazon ECR con un SDK AWS
  - Acciones para Amazon ECR mediante AWS SDKs
    - Úselo CreateRepository con un AWS SDK o CLI
    - Úselo DeleteRepository con un AWS SDK o CLI
    - Úselo DescribeImages con un AWS SDK o CLI
    - Úselo DescribeRepositories con un AWS SDK o CLI
    - Úselo GetAuthorizationToken con un AWS SDK o CLI

- Úselo GetRepositoryPolicy con un AWS SDK o CLI
- Úselo ListImages con un AWS SDK o CLI
- PushImageCmdÚselo con un AWS SDK
- Úselo PutLifeCyclePolicy con un AWS SDK o CLI
- Úselo SetRepositoryPolicy con un AWS SDK o CLI
- Úselo StartLifecyclePolicyPreview con un AWS SDK o CLI

Ejemplos básicos de Amazon ECR mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar los aspectos básicos de Amazon Elastic Container Registry con AWS SDKs.

Ejemplos

- Introducción a Amazon ECR
- Conozca los conceptos básicos de Amazon ECR con un SDK AWS
- Acciones para Amazon ECR mediante AWS SDKs
  - Úselo CreateRepository con un AWS SDK o CLI
  - Úselo DeleteRepository con un AWS SDK o CLI
  - Úselo DescribeImages con un AWS SDK o CLI
  - Úselo DescribeRepositories con un AWS SDK o CLI
  - Úselo GetAuthorizationToken con un AWS SDK o CLI
  - Úselo GetRepositoryPolicy con un AWS SDK o CLI
  - Úselo ListImages con un AWS SDK o CLI
  - PushImageCmdÚselo con un AWS SDK
  - Úselo PutLifeCyclePolicy con un AWS SDK o CLI
  - Úselo SetRepositoryPolicy con un AWS SDK o CLI
  - Úselo StartLifecyclePolicyPreview con un AWS SDK o CLI

Introducción a Amazon ECR

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();
```

```
        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Para obtener detalles sobre la API, consulte listImages en la Referencia de la API de AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }
```

```
    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Para obtener información sobre la API, consulte listImages en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
```

```
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Para obtener más información sobre la API, consulta ListImages en la referencia de la API AWS del SDK for Python (Boto3).

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Conozca los conceptos básicos de Amazon ECR con un SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Cree un repositorio de Amazon ECR.
- Establezca políticas de repositorios.
- Recuperar el repositorio URIs.
- Obtenga tokens de autorización de Amazon ECR.
- Establezca políticas del ciclo de vida para repositorios de Amazon ECR.
- Inserte una imagen de Docker en un repositorio de Amazon ECR.
- Compruebe la existencia de una imagen en un repositorio de Amazon ECR.
- Enumere los repositorios de Amazon ECR de su cuenta y obtenga información acerca de ellos.
- Elimine repositorios de Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecute un escenario interactivo en el que se demuestren las características de Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
```

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact
 with the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text.
 Without a local image,
 * this Java program will not successfully run. For more information including
 how to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions
 to access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }
```

```
        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed
Docker container registry
             service provided by AWS. It allows developers and organizations to
securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides
a set of methods to
             programmatically interact with the Amazon ECR service. This allows
developers to
             automate the storage, retrieval, and management of container images
as part of their application
             deployment pipelines. With ECR, teams can focus on building and
deploying their
             applications without having to worry about the underlying
infrastructure required to
             host and manage a container registry.

            This scenario walks you through how to perform key operations for
this service.
            Let's get started...

            You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                System.out.println("Continuing with the program...");
```

```
                System.out.println("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String repoName = "echo-text";
                ecrActions.deleteECRRepository(repoName);
                return;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("""
           1. Create an ECR repository.

           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it
 easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
```

```
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
```

```
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("An error occurred while retrieving the
authorization token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker
images stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
```

```
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images,
ensuring that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);
```

```
        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker
 image to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the
 image in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS
 CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=
%s
```

```
            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
repository
        or use the force option (used in this scenario) to delete the repository
and have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " +
e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
```

```
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }
   }
}
```

Una clase contenedora para métodos del SDK de Amazon ECR.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
```

```
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
```

```
    * Deletes an ECR (Elastic Container Registry) repository.
    *
    * @param repoName the name of the repository to delete.
    * @throws IllegalArgumentException if the repository name is null or empty.
    * @throws EcrException if there is an error deleting the repository.
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " +
repoName + " repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
               }
           }
       });

       // Wait for the CompletableFuture to complete
       response.join();
   }



   private static DockerClient getDockerClient() {
```

```
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
         and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
         It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
```

```
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
```

```
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
```

```
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null &&
!describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request =
 DescribeRepositoriesRequest.builder()
```

```
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
```

```
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
           }
       });
       response.join();
   }

   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
```

```
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
       });

       GetRepositoryPolicyResponse result = response.join();
       return result != null ? result.policyText() : null;
   }

   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
```

```
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
           {
            "Version" : "2012-10-17",
            "Statement" : [ {
              "Sid" : "new statement",
              "Effect" : "Allow",
              "Principal" : {
                "AWS" : "%s"
              },
              "Action" : "ecr:BatchGetImage"
            } ]
          }
           """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           }
```

```
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
```

```
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
```

```
            return false;
        }
    }
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecute un escenario interactivo en el que se demuestren las características de Amazon ECR.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
 *
 * This code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a
 local image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
```

```
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
        service provided by AWS. It allows developers and organizations to
securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as
part of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
```

```
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-
text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.
```

```
        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
```

```
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
        correct container image from the ECR repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
        These policies allow you to automatically remove old or unused Docker
images from your repositories,
        freeing up storage space and reducing costs.

        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon
ECR by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
image? (y/n)")
```

```
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids
 imageTag=$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
```

```
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Una clase contenedora para métodos del SDK de Amazon ECR.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
```

```
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
```

```
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
```

```
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

```
    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
```

```
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeResponse =
ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
```

```
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }
```

```
            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null &&
!describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
```

```
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }
```

```
            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String =
 repoData?.repositoryUri?.split("/")?.get(0) ?: ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario interactivo en un símbolo del sistema.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations
 using
    Amazon ECR.
    """

    def __init__(
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None

    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
ECR provides a simple and scalable way to manage container images throughout
 their lifecycle,
from building and testing to production deployment.

The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of
 their application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required
 to
host and manage a container registry.
```

```
This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
        print(f"Creating a repository named {self.repository_name}")
        self.repository =
 self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
```

```
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for
 this repository.
            """
            )
        else:
            print(
                """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
the security and integrity of your container images. The repository policy allows
 you to
define specific rules and restrictions for accessing and managing the images
 stored within your ECR
repository.
        """
            )

            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.
```

```
You need an authorization token to securely access and interact with the Amazon
 ECR registry.
The `get_authorization_token` method of the `ecr` client is responsible for
 securely accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any
 operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )

        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image
 URI,
which includes the ECR repository URI. This allows the container runtime to pull
 the
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
        press_enter_to_continue()
        print_dashes()

        print(
            """
```

```
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored
 in your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")

        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
        for line in resp:
            print(line)

        print_dashes()
```

```
        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR
 by using the CLI."
        )
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you
 need to authenticate with the registry. You can do this using the AWS CLI:

    aws ecr get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:

   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )

        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
```

```
        :param ask: If True, prompts the user to confirm before deleting the
resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
'{self.repository_name}? (y/n) "
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the local Docker image
'{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version": "2008-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
```

```
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
 images from a repository.",
        required=False,
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
```

```
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup(False)
```

ECRWrapper clase que envuelve las acciones de Amazon ECR.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
```

```
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
```

```
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.
```

```
        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.
```

```
        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
```

```
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones para Amazon ECR mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo realizar acciones individuales de Amazon ECR con AWS SDKs. Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones para configurar y ejecutar el código.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la Referencia de la API de Amazon Elastic Container Registry.

Ejemplos

- Úselo CreateRepository con un AWS SDK o CLI
- Úselo DeleteRepository con un AWS SDK o CLI
- Úselo DescribeImages con un AWS SDK o CLI
- Úselo DescribeRepositories con un AWS SDK o CLI
- Úselo GetAuthorizationToken con un AWS SDK o CLI
- Úselo GetRepositoryPolicy con un AWS SDK o CLI
- Úselo ListImages con un AWS SDK o CLI
- PushImageCmdÚselo con un AWS SDK
- Úselo PutLifeCyclePolicy con un AWS SDK o CLI
- Úselo SetRepositoryPolicy con un AWS SDK o CLI
- Úselo StartLifecyclePolicyPreview con un AWS SDK o CLI

Úselo **`CreateRepository`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateRepository`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Ejemplo 1: para crear un repositorio

En el siguiente ejemplo de `create-repository` se crea un repositorio dentro del espacio de nombres especificado en el registro predeterminado de una cuenta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Salida:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Para obtener más información, consulte Creating a Repository en la Guía del usuario de Amazon ECR.

Ejemplo 2: creación de un repositorio configurado con inmutabilidad de las etiquetas de imagen

En el siguiente ejemplo de `create-repository` se crea un repositorio configurado para la inmutabilidad de etiquetas en el registro predeterminado de una cuenta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Salida:

```
{
    "repository": {
```

```
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Para obtener más información, consulte Mutabilidad de las etiquetas de imagen en la Guía del usuario de Amazon ECR.

Ejemplo 3: creación de un repositorio configurado con una configuración de escaneo

En el siguiente ejemplo de `create-repository` se crea un repositorio configurado para realizar un escaneo de vulnerabilidad a una inserción de imagen en el registro predeterminado de una cuenta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Salida:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Para obtener más información, consulte Image Scanning en la Guía del usuario de Amazon ECR.

- Para obtener más información sobre la API, consulte CreateRepositoryla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
```

```
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Para obtener más información sobre la API, consulta CreateRepositoryla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
```

```
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws RepositoryAlreadyExistsException if the repository exists.
    * @throws EcrException         if an error occurs while creating the
repository.
    */
   suspend fun createECRRepository(repoName: String?): String? {
       val request =
           CreateRepositoryRequest {
               repositoryName = repoName
           }

       return try {
           EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
               val response = ecrClient.createRepository(request)
               response.repository?.repositoryArn
           }
       } catch (e: RepositoryAlreadyExistsException) {
           println("Repository already exists: $repoName")
           repoName?.let { getRepoARN(it) }
       } catch (e: EcrException) {
           println("An error occurred: ${e.message}")
           null
       }
   }
```

- Para obtener más información sobre la API, consulta CreateRepositorylla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
 self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obtener más información sobre la API, consulta CreateRepositorylа AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteRepository`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteRepository`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Eliminación de un repositorio

En el siguiente ejemplo de `delete-repository` el comando fuerza la eliminación del repositorio especificado del registro predeterminado de una cuenta. La marca `--force` es obligatoria si el repositorio contiene imágenes.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Salida:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
```

```
    }
}
```

Para obtener más información, consulte Deleting a Repository en la Guía del usuario de Amazon ECR.

- Para obtener más información sobre la API, consulte DeleteRepositoryla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Deletes an ECR (Elastic Container Registry) repository.
    *
    * @param repoName the name of the repository to delete.
    * @throws IllegalArgumentException if the repository name is null or empty.
    * @throws EcrException if there is an error deleting the repository.
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
```

```
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " +
repoName + " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- Para obtener más información sobre la API, consulta DeleteRepositoryla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 */
suspend fun deleteECRRepository(repoName: String) {
    if (repoName.isNullOrEmpty()) {
```

```
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Para obtener más información sobre la API, consulta DeleteRepositoryla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
```

```
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteRepositoryla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeImages`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeImages`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Para describir una imagen en un repositorio

En el siguiente ejemplo de `describe-images` se muestran detalles sobre una imagen del repositorio `cluster-autoscaler` con la etiqueta `v1.13.6`.

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Salida:

```
{
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
}
```

- Para obtener más información sobre la API, consulte DescribeImagesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
```

```
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Para obtener más información sobre la API, consulta DescribeImagesla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
```

```
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para obtener más información sobre la API, consulta DescribeImagesla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DescribeImagesla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeRepositories`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeRepositories`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Para describir los repositorios de imágenes en un registro

En este ejemplo, se describen los repositorios del registro predeterminado de una cuenta.

Comando:

```
aws ecr describe-repositories
```

Salida:

```
{
    "repositories": [
        {
            "registryId": "012345678910",
            "repositoryName": "ubuntu",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
        },
        {
```

```
            "registryId": "012345678910",
            "repositoryName": "test",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
        }
    ]
}
```

- Para obtener más información sobre la API, consulte DescribeRepositoriesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
```

```
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- Para obtener más información sobre la API, consulta DescribeRepositoriesla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Para obtener más información sobre la API, consulta DescribeRepositoriesla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- Para obtener más información sobre la API, consulta DescribeRepositoriesla AWS Referencia de API de SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
 aws_sdk_ecr::Error> {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- Para obtener más información sobre la API, consulta DescribeRepositoriesla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon ECR con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetAuthorizationToken`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetAuthorizationToken`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Para obtener un token de autorización para su registro predeterminado

En el siguiente ejemplo de comando de `get-authorization-token` se obtiene un token de autorización para el registro predeterminado.

```
aws ecr get-authorization-token
```

Salida:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Para obtener más información sobre la API, consulte GetAuthorizationTokenla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
```

```
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
```

- Para obtener más información sobre la API, consulta GetAuthorizationTokenla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- Para obtener más información sobre la API, consulta GetAuthorizationTokenla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta GetAuthorizationTokenla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetRepositoryPolicy`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetRepositoryPolicy`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Para recuperar la política de repositorio de un repositorio

En el siguiente ejemplo de `get-repository-policy` se muestran detalles sobre la política de repositorio del repositorio `cluster-autoscaler`.

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Salida:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
```

```
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Para obtener más información sobre la API, consulte GetRepositoryPolicyla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
```

```
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- Para obtener más información sobre la API, consulta GetRepositoryPolicyla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
```

```
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
 ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- Para obtener más información sobre la API, consulta GetRepositoryPolicyla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
```

```
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obtener más información sobre la API, consulta GetRepositoryPolicyla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListImages`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListImages`.

CLI

AWS CLI

Para crear una lista de las imágenes en un repositorio

En el siguiente ejemplo de `list-images` se muestra una lista de las imágenes del repositorio `cluster-autoscaler`.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Salida:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
        }
    ]
}
```

- Para obtener más información sobre la API, consulte ListImagesla Referencia de AWS CLI comandos.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();

    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- Para obtener más información sobre la API, consulta ListImagesla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon ECR con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

PushImageCmdÚselo con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `PushImageCmd`.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName  the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   public void pushDockerImage(String repoName, String imageName) {
       System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
       CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
           .thenApply(response -> {
               String token =
response.authorizationData().get(0).authorizationToken();
               String decodedToken = new
String(Base64.getDecoder().decode(token));
               String password = decodedToken.substring(4);

               DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
               Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
```

```
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Para obtener más información sobre la API, consulta PushImageCmdla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
```

```
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- Para obtener más información sobre la API, consulta PushImageCmdla referencia sobre el AWS SDK para la API de Kotlin.

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaUso de Amazon ECR con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`PutLifeCyclePolicy`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `PutLifeCyclePolicy`.

CLI

AWS CLI

Para crear una política de ciclo de vida

En el siguiente ejemplo de `put-lifecycle-policy`,se crea una política de ciclo de vida para el repositorio especificado en el registro predeterminado de una cuenta.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
```

```
    --lifecycle-policy-text "file://policy.json"
```

Contenido de `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
               "type": "expire"
           }
       }
   ]
}
```

Salida:

```
{
   "registryId": "<aws_account_id>",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\"rules\":[{\"rulePriority\":1,\"description\":
\"Expire images older than 14 days\",\"selection\":{\"tagStatus\":\"untagged\",
\"countType\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},
\"action\":{\"type\":\"expire\"}}]}"
}
```

Para obtener más información, consulte Lifecycle Policies en la Guía del usuario de Amazon ECR.

- Para obtener más información sobre la API, consulte PutLifeCyclePolicyla Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
 str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
 policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
```

```
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Ejemplo que pone una política de fecha de caducidad.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Para obtener más información sobre la API, consulta PutLifeCyclePolicyla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`SetRepositoryPolicy`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SetRepositoryPolicy`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Para establecer la política de repositorio de un repositorio

En el siguiente ejemplo de `set-repository-policy`, se asocia al repositorio `cluster-autoscaler` una política de repositorio incluida en un archivo.

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Contenido de `my-policy.json`:

```
{
    "Version" : "2008-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
```

```
    ]
}
```

Salida:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Para obtener más información sobre la API, consulte SetRepositoryPolicyla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
```

```
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
           {
            "Version" : "2012-10-17",
            "Statement" : [ {
              "Sid" : "new statement",
              "Effect" : "Allow",
              "Principal" : {
                "AWS" : "%s"
              },
              "Action" : "ecr:BatchGetImage"
            } ]
          }
           """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           }
       });
       response.join();
```

```
    }
```

- Para obtener más información sobre la API, consulta SetRepositoryPolicyla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
         {
          "Version" : "2012-10-17",
          "Statement" : [ {
            "Sid" : "new statement",
            "Effect" : "Allow",
            "Principal" : {
              "AWS" : "$iamRole"
            },
            "Action" : "ecr:BatchGetImage"
          } ]
        }
```

```
            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Para obtener más información sobre la API, consulta SetRepositoryPolicyla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
```

```
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Ejemplo que concede a un rol de IAM acceso de descarga.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
```

```
            "Version": "2008-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Para obtener más información sobre la API, consulta SetRepositoryPolicyla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Amazon ECR con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`StartLifecyclePolicyPreview`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `StartLifecyclePolicyPreview`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

CLI

AWS CLI

Para crear una vista previa de política de ciclo de vida

En el siguiente ejemplo de `start-lifecycle-policy-preview` se crea una vista previa de la política de ciclo de vida definida por un archivo JSON para el repositorio especificado.

```
aws ecr start-lifecycle-policy-preview \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Contenido de `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
               "type": "expire"
           }
       }
   ]
}
```

Salida:

```
{
   "registryId": "012345678910",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\n    \"rules\": [\n        {\n
 \"rulePriority\": 1,\n            \"description\": \"Expire images older than 14
 days\",\n            \"selection\": {\n                \"tagStatus\": \"untagged
\",\n                \"countType\": \"sinceImagePushed\",\n
 \"countUnit\": \"days\",\n                \"countNumber\": 14\n            },\n
           \"action\": {\n                \"type\": \"expire\"\n            }\n
      }\n    ]\n}\n",
   "status": "IN_PROGRESS"
}
```

- Para obtener más información sobre la API, consulte StartLifecyclePolicyPreviewla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
```

```
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Para obtener más información sobre la API, consulta StartLifecyclePolicyPreviewla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
```

```
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para obtener más información sobre la API, consulta StartLifecyclePolicyPreviewla referencia sobre el AWS SDK para la API de Kotlin.

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaUso de Amazon ECR con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cuotas de servicio de Amazon ECR

En la tabla siguiente se proporcionan las cuotas de servicio predeterminadas de Amazon Elastic Container Registry (Amazon ECR).

Nombre	Valor predeterminado	Ajusta	Descripción
Escaneos de imágenes básicos cada 24 horas	Cada región admitida: 100 000	No	El número máximo de imágenes que se pueden escanear en un período de 24 horas en la cuenta corriente y la región mediante el escaneo básico. Este límite incluye tanto el escaneado automático como el manual.
Filtros por regla en una configuración de replicación	Cada región admitida: 100	No	El número máximo de filtros por regla en una configuración de replicación.
Imágenes por repositorio	Cada región admitida: 100 000	Sí	El número máximo de imágenes por repositorio.
Partes de capa	Cada región admitida: 4200	No	El número máximo de partes de capa. Esto solo es aplicable si utiliza las acciones de la API de Amazon ECR directamente para comenzar cargas multiparte para operaciones de inserción de imágenes.

Nombre	Valor predeterminado	Ajusta	Descripción
Duración de la política del ciclo de vida	Cada región admitida: 30 720	No	Número máximo de caracteres en una política de ciclo de vida.
Tamaño máximo de parte de la capa	Cada región admitida: 10	No	Tamaño máximo (MiB) de una parte de capa. Esto solo es aplicable si utiliza las acciones de la API de Amazon ECR directamente para comenzar cargas multiparte para operaciones de inserción de imágenes.
Tamaño máximo de capa	Cada región admitida: 52 000	No	Tamaño máximo (MiB) de una capa.
Tamaño mínimo de parte de la capa	Cada región admitida: 5	No	Tamaño mínimo (MiB) de una parte de capa. Esto solo es aplicable si utiliza las acciones de la API de Amazon ECR directamente para comenzar cargas multiparte para operaciones de inserción de imágenes.
Reglas de caché de extracción por registro	Cada región admitida: 50	No	Número máximo de reglas de caché de extracción.

Nombre	Valor predeterminado	Ajusta	Descripción
Porcentaje de BatchCheckLayerAvailability solicitudes	Cada región admitida: 1000 por segundo	Sí	El número máximo de BatchCheckLayerAvailability solicitudes que puede realizar por segundo en la región actual. Cuando se inserta una imagen en un repositorio, se comprueba cada capa de la imagen para verificar si se ha cargado antes. Si ya se ha cargado, se omite.
Tasa de BatchGetImage solicitudes	Cada región admitida: 2000 por segundo	Sí	El número máximo de BatchGetImage solicitudes que puede realizar por segundo en la región actual. Cuando se extrae una imagen, se llama a la BatchGetImage API una vez para recuperar el manifiesto de la imagen. Si solicitas un aumento de la cuota de esta API, revisa también tu GetDownloadUrlForLayer uso.

Nombre	Valor predeterminado	Ajusta	Descripción
Tasa de CompleteLayerUpload solicitudes	Cada región admitida: 100 por segundo	Sí	El número máximo de CompleteLayerUpload solicitudes que puede realizar por segundo en la región actual. Cuando se inserta una imagen, se llama a la CompleteLayerUpload API una vez por cada nueva capa de imagen para comprobar que la carga se ha completado.
Tasa de GetAuthorizationToken solicitudes	Cada región admitida: 500 por segundo	Sí	El número máximo de GetAuthorizationToken solicitudes que puede realizar por segundo en la región actual.
Tasa de GetDownloadUrlForLayer solicitudes	Cada región admitida: 3000 por segundo	Sí	El número máximo de GetDownloadUrlForLayer solicitudes que puede realizar por segundo en la región actual. Cuando se extrae una imagen, se llama a la GetDownloadUrlForLayer API una vez por capa de imagen que aún no esté almacenada en caché. Si solicitas un aumento de la cuota de esta API, revisa también tu BatchGetImage uso.

Nombre	Valor predeterminado	Ajusta	Descripción
Tasa de InitiateLayerUpload solicitudes	Cada región admitida: 100 por segundo	Sí	El número máximo de InitiateLayerUpload solicitudes que puede realizar por segundo en la región actual. Cuando se inserta una imagen, se llama a la InitiateLayerUpload API una vez por capa de imagen que aún no se haya cargado. La acción de la BatchCheckLayerAvailability API determina si se ha cargado o no una capa de imágenes.
Tasa de PutImage solicitudes	Cada región admitida: 10 por segundo	Sí	El número máximo de PutImage solicitudes que puede realizar por segundo en la región actual. Cuando se inserta una imagen y se cargan todas las capas de imágenes nuevas, se invoca a la PutImage API una vez para crear o actualizar el manifiesto de la imagen y las etiquetas asociadas a la imagen.

Nombre	Valor predeterminado	Ajusta	Descripción
Tasa de UploadLayerPart solicitudes	Cada región admitida: 500 por segundo	Sí	El número máximo de UploadLayerPart solicitudes que puede realizar por segundo en la región actual. Cuando se inserta una imagen, cada nueva capa de imagen se carga en partes y se llama a la UploadLayerPart API una vez por cada nueva parte de la capa de imagen.
Tasa de escaneos de imágenes	Cada región admitida: 1	No	Número máximo de escaneos de imágenes por imagen cada 24 horas.
Repositorios registrados	Cada región admitida: 100 000	Sí	El número máximo de repositorios que puede crear en esta cuenta en la región actual.
Reglas por política de ciclo de vida	Cada región admitida: 50	No	El número máximo de reglas en una política de ciclo de vida
Reglas por configuración de replicación	Cada región admitida: 10	No	El número máximo de reglas en una configuración de replicación.
Etiquetas por imagen	Cada región admitida: 1000	No	El número máximo de etiquetas por imagen.

Nombre	Valor predeterminado	Ajusta	Descripción
Destinos únicos en todas las reglas de una configuración de replicación	Cada región admitida: 25	No	El número máximo de destinos únicos en todas las reglas de una configuración de replicación.

Administración de las cuotas de servicio de Amazon ECR en la AWS Management Console

Amazon ECR se ha integrado con Service Quotas, un AWS servicio que le permite ver y gestionar sus cuotas desde una ubicación central. Para obtener más información, consulte ¿Qué son las Service Quotas? en la Guía del usuario de Service Quotas.

Con Service Quotas, resulta más sencillo buscar el valor de todas las cuotas de servicio de Amazon ECR.

Visualización de las cuotas de servicio de Amazon ECR (AWS Management Console)

1. Abra la consola de Service Quotas en https://console.aws.amazon.com/servicequotas/.
2. En el panel de navegación, elija AWS services (Servicios de AWS).
3. En la lista AWS services (Servicios de AWS), busque y seleccione Amazon Elastic Container Registry (Amazon ECR).

   En la lista de cuotas de servicio, puede ver el nombre de la cuota de servicio, el valor aplicado (si está disponible), la cuota AWS predeterminada y si el valor de la cuota es ajustable.
4. Para ver información adicional sobre una cuota de servicio, como, por ejemplo, la descripción, elija el nombre de cuota.

Para solicitar un aumento de cuota, consulte Solicitud de un aumento de cuota en la Guía de usuario de Service Quotas.

Crear una CloudWatch alarma para monitorear las métricas de uso de la API

Amazon ECR proporciona métricas de CloudWatch uso que corresponden a las cuotas de AWS servicio de cada una de las partes APIs implicadas en las acciones de autenticación del registro, inserción y extracción de imágenes. En la consola de Service Quotas, puede visualizar la utilización en gráficos y configurar alarmas que le avisen cuando este se aproxime a una cuota de servicio. Para obtener más información, consulte Métricas de uso de Amazon ECR.

Siga los siguientes pasos para crear una CloudWatch alarma basada en una de las métricas de uso de la API Amazon ECR.

Creación de una alarma basada en las cuotas de uso de Amazon ECR (AWS Management Console)

1. Abra la consola de Service Quotas en https://console.aws.amazon.com/servicequotas/.
2. En el panel de navegación, elija AWS services (Servicios de AWS).
3. En la lista AWS services (Servicios de AWS), busque y seleccione Amazon Elastic Container Registry (Amazon ECR).
4. En la lista Service quotas (Cuotas de servicio), seleccione la cuota de uso de Amazon ECR para la que desee crear una alarma.
5. En la sección de alarmas de Amazon CloudWatch Events, selecciona Crear.
6. En Alarm threshold (Umbral de alarma), elija el porcentaje del valor de la cuota aplicada que desee establecer como valor de la alarma.
7. En Alarm name (Nombre de la alarma), escriba el nombre de la alarma y elija Create (Crear).

Solución de problemas de Amazon ECR

En este capítulo se proporciona ayuda para encontrar información de diagnóstico de Amazon ECR y pasos para resolver problemas y saber cómo actuar con mensajes de error comunes.

Temas

- Solución de problemas con los comandos de Docker y errores al utilizar Amazon ECR
- Solución de problemas de los mensajes de error de Amazon ECR

Solución de problemas con los comandos de Docker y errores al utilizar Amazon ECR

En algunos casos, al ejecutar un comando de Docker en Amazon ECR puede aparecer un mensaje de error. A continuación se explican algunos de los mensajes de error más comunes y sus posibles soluciones.

Temas

- Los registros de Docker no contienen mensajes de error esperados
- Error «Filesystem Verification Failed» (Fallo al verificar el sistema de archivos) o «404: Image Not Found» (Imagen no encontrada) al extraer una imagen de un repositorio de Amazon ECR
- Error «Filesystem Layer Verification Failed» (Fallo al verificar la capa del sistema de archivos) al extraer imágenes de Amazon ECR
- Errores HTTP 403 o error «no basic auth credentials» (no hay credenciales de autenticación básica) al insertar contenido en un repositorio

Los registros de Docker no contienen mensajes de error esperados

Para comenzar a solucionar cualquier problema relacionado con Docker, active primero el resultado de depuración del daemon de Docker en las instancias de host. Si utiliza imágenes extraídas de Amazon ECR en instancias de contenedor de Amazon ECS, consulte Configurar el resultado detallado del daemon de Docker en la Guía para desarrolladores de Amazon Elastic Container Service.

Error «Filesystem Verification Failed» (Fallo al verificar el sistema de archivos) o «404: Image Not Found» (Imagen no encontrada) al extraer una imagen de un repositorio de Amazon ECR

Puede producirse el error `Filesystem verification failed` al ejecutar el comando docker pull para extraer una imagen de un repositorio de Amazon ECR con Docker 1.9 o versiones posteriores. Es posible que reciba el error `404: Image not found` cuando utilice versiones de Docker anteriores a la 1.9.

A continuación se explican los posibles motivos.

El disco local está lleno

- Si el disco local en el que ejecuta el comando docker pull está lleno, es posible que el hash SHA-1 que se haya calculado en el archivo local sea distinto del que haya calculado Amazon ECR. Compruebe que el disco local cuenta con suficiente espacio libre para almacenar la imagen de Docker que está extrayendo. Puede eliminar imágenes antiguas y así hacer sitio para las nuevas. Ejecute el comando docker images para ver una lista de las imágenes de Docker descargadas localmente y sus tamaños.

El cliente no se puede conectar al repositorio remoto debido a un error de red.

- Las llamadas a un repositorio de Amazon ECR requieren conexión a Internet. Verifique la configuración de red y compruebe que otras herramientas y aplicaciones sí pueden obtener acceso a recursos en Internet. Si utilizas una EC2 instancia de Amazon en una subred privada, comprueba que la subred tenga una ruta a Internet. docker pull Utilice un servidor de conversión de las direcciones de red (NAT) o una gateway de NAT administrada.

  Actualmente, las llamadas a un repositorio de Amazon ECR también requieren acceso de red a Amazon Simple Storage Service (Amazon S3) a través del firewall de su compañía. Si su organización utiliza software de firewall o un dispositivo NAT que permite puntos de enlace de servicio, asegúrese de que los puntos de enlace de servicio de Amazon S3 de su región actual estén permitidos.

  Si está usando Docker tras un proxy HTTP, puede configurar Docker con los ajustes del proxy correspondientes. Para obtener más información, consulte la sección HTTP proxy de la documentación de Docker.

Error «Filesystem Layer Verification Failed» (Fallo al verificar la capa del sistema de archivos) al extraer imágenes de Amazon ECR

Puede que se produzca el error `image image-name not found` al usar el comando docker pull para extraer imágenes. Si revisa los logs de Docker, es posible que encuentre un error como el siguiente:

```
filesystem layer verification failed for digest sha256:2b96f...
```

Este error indica que no se han podido descargar una o varias capas de la imagen. A continuación se explican los posibles motivos.

Está usando una versión antigua de Docker

Este error puede ocurrir en un reducido porcentaje de casos al utilizar una versión de Docker anterior a la 1.10. Actualice Docker a la versión 1.10 o a una posterior.

El cliente ha detectado un error de red o disco

Un disco lleno o un problema de red pueden impedir la descarga de una o varias capas, tal como se ha explicado previamente en la descripción del mensaje `Filesystem verification failed`. Siga las recomendaciones anteriores para asegurarse de que el sistema de archivos no está lleno y que la red permite el acceso a Amazon S3.

Errores HTTP 403 o error «no basic auth credentials» (no hay credenciales de autenticación básica) al insertar contenido en un repositorio

Hay ocasiones en que es posible que reciba un error `HTTP 403 (Forbidden)` o el mensaje de error `no basic auth credentials` desde los comandos docker push o docker pull, incluso a pesar de haberse autenticado correctamente en Docker con el comando aws ecr get-login-password. A continuación se indican algunas causas conocidas de este problema:

Se ha autenticado en una región diferente

Las solicitudes de autenticación están vinculadas con regiones específicas y no se pueden utilizar en otras regiones. Por ejemplo, si obtiene un token de autorización de EE. UU. Oeste (Oregón), no puede utilizarlo para autenticarse en sus repositorios en EE. UU. Este (Norte de Virginia). Para resolver el problema, asegúrese de haber recuperado un token de autenticación de la misma

región en la que se encuentra el repositorio. Para obtener más información, consulte the section called “Autenticación del registro”.

Se ha autenticado para enviar a un repositorio para el que no tiene permisos

No dispone de los permisos necesarios para enviar al repositorio. Para obtener más información, consulte Políticas de repositorios privados en Amazon ECR.

Su token ha caducado.

El periodo predeterminado de vencimiento de los tokens de autorización obtenidos mediante la operación `GetAuthorizationToken` es de 12 horas.

Error en el gestor de credenciales `wincred`

Algunas versiones de Docker para Windows utilizan un administrador de credenciales llamado`wincred`, que no gestiona correctamente el comando de inicio de sesión de Docker generado por aws ecr get-login-password (para obtener más información, consulta la sección Errores en los repositorios privados) `CredsStore`. Puede ejecutar el comando de inicio de sesión de Docker generado, pero si intenta enviar o extraer imágenes, el comando falla. Este problema se puede solucionar eliminando el esquema `https://` del argumento del registro en el comando de inicio de sesión de Docker generado desde aws ecr get-login-password. A continuación se muestra un ejemplo de comando de inicio de sesión de Docker sin el esquema HTTPS.

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Solución de problemas de los mensajes de error de Amazon ECR

En algunos casos, las llamadas a la API que haya desencadenado a través de la consola de Amazon ECR o la AWS CLI generan un mensaje de error. A continuación se explican algunos de los mensajes de error más comunes y sus posibles soluciones.

HTTP 429: demasiadas solicitudes o ThrottleException

Es posible que una o varias acciones de Amazon ECR, o llamadas a la API, generen un error `429: Too Many Requests` o `ThrottleException`. Esto indica que está realizando repetidamente una llamada a un único punto de enlace en Amazon ECR en poco tiempo y que se están limitando sus solicitudes. La limitación controlada ocurre cuando el número de llamadas que realiza un usuario a un único punto de enlace supera una determinada cantidad en un periodo establecido.

Cada operación de API en Amazon ECR tiene un regulador de velocidad asociado. Por ejemplo, la limitación para la acción `GetAuthorizationToken` es de 20 transacciones por segundo (TPS), con una ráfaga máxima permitida de hasta 200 TPS. En cada región, cada cuenta recibe un bucket que puede almacenar hasta 200 créditos de `GetAuthorizationToken`. Estos créditos se reaprovisionan a la velocidad de 20 por segundo. Si su bucket tiene 200 créditos, podría realizar hasta 200 transacciones de API `GetAuthorizationToken` por segundo durante un segundo, y luego 20 transacciones por segundo de forma indefinida. Para obtener más información sobre los límites de tasa de Amazon ECR APIs, consulteCuotas de servicio de Amazon ECR.

Para gestionar errores de limitación controlada, implemente una función de reintento con retardo exponencial en el código. Para obtener más información, consulte el comportamiento de los reintentos en la Guía de referencia de herramientas AWS SDKs y herramientas. Otra opción es solicitar un aumento del límite de velocidad, lo que se puede hacer mediante la consola Service Quotas. Para obtener más información, consulte Administración de las cuotas de servicio de Amazon ECR en la AWS Management Console..

HTTP 403: "User [arn] is not authorized to perform [operación]"

Es posible que aparezca el siguiente error al intentar ejecutar una acción con Amazon ECR:

```
$ aws ecr get-login-password
A client error (AccessDeniedException) occurred when calling the GetAuthorizationToken
 operation:
    User: arn:aws:iam::account-number:user/username is not authorized to perform:
    ecr:GetAuthorizationToken on resource: *
```

Esto indica que al usuario no se le han concedido permisos para utilizar Amazon ECR o que los permisos no están configurados correctamente. Si está realizando acciones en un repositorio de Amazon ECR, compruebe que el usuario tiene permisos de acceso a dicho repositorio. Para obtener más información sobre cómo crear y comprobar permisos para Amazon ECR, consulte Identity and Access Management para Amazon Elastic Container Registry.

Error HTTP 404: «Repository Does Not Exist» (El repositorio no existe)

Si especifica un repositorio de Docker Hub que no existe actualmente, Docker Hub lo crea de forma automática. En Amazon ECR, es necesario crear explícitamente los nuevos repositorios para poder utilizarlos. Esto impide que se creen nuevos repositorios por accidente (por ejemplo, debido a errores ortográficos) y también permite asegurarse de que a cada nuevo repositorio se le asigne

explícitamente una política de acceso de seguridad adecuada. Para obtener más información sobre la creación de repositorios, consulte Repositorios privados de Amazon ECR.

Error: No se puede realizar un inicio de sesión interactivo desde un dispositivo que no sea TTY

Si recibes el error `Cannot perform an interactive login from a non TTY device`, los siguientes pasos de resolución de problemas deberían ayudarle.

- Compruebe que está utilizando la AWS CLI versión 2 y que no tiene una versión conflictiva de la AWS CLI versión 1 en su sistema. Para obtener más información, consulte Instalación o actualización de la versión de AWS CLI más reciente.
- Compruebe que la ha configurado AWS CLI con credenciales válidas. Para obtener más información, consulte Instalación o actualización de la versión de AWS CLI más reciente.
- Compruebe que la sintaxis del AWS CLI comando es correcta.

Uso de Podman con Amazon ECR

El uso de Podman con Amazon ECR permite a las organizaciones aprovechar la seguridad y la simplicidad de Podman y, al mismo tiempo, beneficiarse de la escalabilidad y la fiabilidad de Amazon ECR para la administración de imágenes de contenedores. Al seguir los pasos y comandos descritos, los desarrolladores y administradores pueden optimizar los flujos de trabajo de contenedores, mejorar la seguridad y optimizar la utilización de los recursos. A medida que la creación de contenedores sigue ganando impulso, el uso de Podman y Amazon ECR proporciona una solución sólida y flexible para administrar e implementar aplicaciones en contenedores.

Uso de Podman para la autenticación con Amazon ECR

Antes de interactuar con Amazon ECR por medio de Podman, se requiere autenticación. Para esto, puede ejecutar el comando `aws ecr get-login-password` con el fin de recuperar un token de autenticación y, a continuación, usar ese token con el comando `podman login` para autenticarse con Amazon ECR.

```
aws ecr get-login-password --region region | podman login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Uso del asistente de credenciales Amazon ECR con Podman

Amazon ECR proporciona un asistente de credenciales de Docker que funciona con Podman. El asistente de credenciales facilita el almacenamiento y el uso de las credenciales de Docker al enviar y extraer imágenes a Amazon ECR. Para obtener información sobre los pasos de instalación y configuración, consulte Amazon ECR Docker Credential Helper.

> **Note**
>
> El asistente de credenciales de Docker de Amazon ECR no admite actualmente la autenticación multifactor (MFA).

Extracción de imágenes desde Amazon ECR mediante Podman

Una vez finalizada la autenticación, las imágenes del contenedor se pueden extraer de Amazon ECR mediante el comando `podman pull` con el URI completo del repositorio de Amazon ECR.

```
podman pull aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Ejecución de contenedores para Amazon ECR con Podman

Una vez extraída la imagen deseada, se puede crear una instancia de contenedor mediante el comando `podman run`.

```
podman run -d aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Inserción de imágenes en Amazon ECR con Podman

Para insertar una imagen local en Amazon ECR, primero se debe etiquetar la imagen con el URI del repositorio de Amazon ECR con la ayuda de `podman tag` y, a continuación, usar el comando `podman push` para cargar la imagen en Amazon ECR.

```
podman
 tag local_image:tag aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
podman push aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Historial de documentos

En la siguiente tabla se describen los cambios importantes que se han realizado en la documentación desde la última versión de Amazon ECR. Actualizamos la documentación con frecuencia para dar respuesta a los comentarios que se nos envía.

Cambio	Descripción	Fecha
Se actualizó para incluir compatibilidad con los patrones de exclusión por inmutabilidad de las etiquetas de imagen	Amazon ECR ha actualizado las capacidades de etiquetado de imágenes para incluir patrones de exclusión de la inmutabilidad de las etiquetas de imagen al crear y actualizar los repositorios. Ahora puede especificar etiquetas que se puedan actualizar incluso cuando la inmutabilidad de las etiquetas esté habilitada en un repositorio. Para ello, defina patrones comodín (como `latest` «,,»`dev-*`) para excluir etiquetas específicas de las reglas de inmutabilidad `yv*.beta`, al mismo tiempo, mantener la inmutabil idad de todas las demás etiquetas. Para obtener más información, consulte Creación de un repositorio privado de Amazon ECR para almacenar imágenes.	23 de julio de 2025
Se actualizó el escaneo de imágenes mejorado para proporcionar información sobre el uso de las imágenes	Amazon ECR ha actualizado las capacidades de escaneo de imágenes mejorado para incluir la visibilid ad de cómo se utilizan las imágenes en Amazon EKS y Amazon ECS. Para obtener más información, consulte Escanear imágenes para detectar vulnerabi lidades de paquetes de lenguajes de programación y sistemas operativos en Amazon ECR.	16 de junio de 2025
IPv6 apoyo	Se agregó soporte para realizar solicitudes a los registros de Amazon ECR mediante puntos de enlace de pila IPv4 única y de doble pila (IPv4 and). IPv6 Para obtener más información, consulte Realizar solicitudes a los registros de Amazon ECR.	30 de abril de 2025

Cambio	Descripción	Fecha
Se agregó compatibilidad con el registro privado Amazon ECR para extraer la memoria caché	Amazon ECR agregó soporte para crear reglas de caché de extracción para el registro privado de Amazon ECR. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR y Rol vinculado a servicios de Amazon ECR para la caché de extracción.	12 de marzo de 2025
Se agregó soporte para establecer el alcance de la política de registro	Amazon ECR agregó soporte para configurar el alcance de la política de registro para su registro privado. Para obtener más información, consulte Permisos de registro privado en Amazon ECR y Registro privado de Amazon ECR.	23 de diciembre de 2024
Amazon EC2 Container RegistryPullOnly — Nueva política	Amazon ECR agregó una nueva política que otorga permisos de solo extracción a Amazon ECR.	10 de octubre de 2024
Las operaciones mediante proxy de clientes de Docker/OCI en los eventos ahora apuntan a CloudTrail `ecr.amazonaws.com`	El valor `ecr.amazonaws.com` reemplaza a los campos Agente `AWS Internal` de usuario (`userAgent`) y Dirección IP de origen () para los eventos asociados a los puntos `sourceIPAddress` finales del cliente. CloudTrail Docker/OCI Para ver ejemplos, consulte Ejemplo: Acción de extracción de imágenes y Ejemplo: Acción de inserción de imágenes.	1 de julio de 2024
Se añadió una descripción al nuevo rol vinculado a servicios de Amazon ECR para plantillas de creación de repositorios.	Amazon ECR utiliza un rol vinculado a un servicio denominado AWSServiceRoleForECRTemplateque permite a Amazon ECR realizar acciones en su nombre para completar las acciones de la plantilla de creación de repositorios. Para obtener más información, consulte Rol vinculado a servicios de Amazon ECR para plantillas de creación de repositorios.	20 de junio de 2024

Cambio	Descripción	Fecha
Se ha agregado el rol vinculado al servicio de `ECRTemplateServiceRolePolicy` .	Se ha agregado el rol vinculado al servicio de `ECRTemplateServiceRolePolicy` . Para obtener más información, consulte ECRTemplateServiceRolePolicy	20 de junio de 2024
Se añadió la replicación entre regiones y entre cuentas en las regiones de China.	Amazon ECR ha añadido compatibilidad a la región de China para filtrar los repositorios que se replican. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR	15 de mayo de 2024
Se agregó GitLab un registro de contenedores para consultar las reglas de caché	Amazon ECR agregó soporte para crear reglas de caché de extracción para el registro de GitLab contenedores. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.	8 de mayo de 2024
Actualización de la política de ciclo de vida de Amazon ECR para añadir compatibilidad con el uso de comodines	Amazon ECR agregó compatibilidad con comodines en una política de ciclo de vida mediante el uso del parámetro `tagPatternList` en una regla de política de ciclo de vida. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.	18 de diciembre de 2023
Plantillas de creación de repositorios de Amazon ECR	Amazon ECR agregó compatibilidad para las plantillas de creación de repositorios. Para obtener más información, consulte Plantillas para controlar los repositorios creados durante una acción de extracción, caché o replicación.	15 de noviembre de 2023
Se agregó la caché de extracción de Amazon ECR, compatible con los registros ascendentes autenticados	Amazon ECR agregó compatibilidad para el uso de registros ascendentes que requieren autenticación para sus reglas de caché de extracción. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.	15 de noviembre de 2023

Cambio	Descripción	Fecha
AWSECRPullThroughCache_ServiceRolePolicy: actualización de una política actual	Amazon ECR agregó nuevos permisos a la política `AWSECRPullThroughCache_ServiceRolePolicy` . Estos permisos permiten a Amazon ECR recuperar el contenido cifrado de un secreto de Secrets Manager. Esto es necesario cuando se utiliza una regla de caché de extracción para almacenar en caché las imágenes de un registro anterior que requiere autenticación.	15 de noviembre de 2023
Firma de imágenes de Amazon ECR	Amazon ECR y AWS Signer se agregó soporte para crear y enviar firmas de imágenes en contenedores mediante el cliente Notary. Para obtener más información, consulte Firma de una imagen almacenada en un repositorio privado de Amazon ECR.	6 de junio de 2023
Se incorporó el registro de contenedores de Kubernetes a las reglas de caché de extracción	Amazon ECR incorporó compatibilidad para crear reglas de caché de extracción para el registro de contenedores de Kubernetes. Para obtener más información, consulte Sincronización de un registro principal con un registro privado de Amazon ECR.	1 de junio de 2023
Compatibilidad con duración de análisis mejorado de Amazon ECR	Amazon Inspector ha agregado compatibilidad para establecer el tiempo durante el que se supervisan sus repositorios cuando se habilita el análisis mejorado. Para obtener más información, consulte Cambio de la duración del escaneo mejorado de imágenes en Amazon Inspector.	28 de junio de 2022
Amazon ECR envía las métricas del recuento de extracciones del repositorio a Amazon CloudWatch	Amazon ECR envía las métricas del recuento de extracciones del repositorio a Amazon CloudWatch. Para obtener más información, consulte Métricas de repositorios de Amazon ECR.	6 de enero de 2022

Cambio	Descripción	Fecha
Compatibilidad ampliada con la replicación	Amazon ECR ha agregado compatibilidad para filtrar los repositorios que se replican. Para obtener más información, consulte Replicación de imágenes privadas en Amazon ECR.	21 de septiembre de 2021
AWS políticas gestionadas para Amazon ECR	Amazon ECR ha añadido documentación sobre las políticas AWS gestionadas. Para obtener más información, consulte AWS políticas gestionadas para Amazon Elastic Container Registry.	24 de junio de 2021
Replicación entre regiones y entre cuentas	Amazon ECR ha agregado compatibilidad con objeto de configurar los parámetros de replicación para su registro privado. Para obtener más información, consulte Configuración del registro privado en Amazon ECR.	8 de diciembre de 2020
Compatibilidad con los artefactos de OCI	Amazon ECR ha agregado compatibilidad con la inserción y extracción de artefactos de Open Container Initiative (OCI). Se ha agregado un nuevo parámetro `artifactMediaType` a la respuesta de la API `DescribeImages` para indicar el tipo de artefacto. Para obtener más información, consulte Inserción de un gráfico de Helm en un repositorio privado de Amazon ECR.	24 de agosto de 2020
Cifrado en reposo	Amazon ECR ha agregado compatibilidad para configurar el cifrado de sus repositorios mediante el cifrado del lado del servidor con claves administradas por el cliente almacenadas en AWS Key Management Service (AWS KMS). Para obtener más información, consulte Cifrado en reposo.	29 de julio de 2020

Cambio	Descripción	Fecha
Imágenes multiarquitectura	Amazon ECR ha agregado compatibilidad con objeto de crear e insertar listas de manifiesto Docker que se utilizan para imágenes multiarquitectura. Para obtener más información, consulte Inserción de una imagen multiarquitectura a un repositorio privado de Amazon ECR.	28 de abril de 2020
Métricas de uso de Amazon ECR	Amazon ECR agregó métricas CloudWatch de uso que proporcionan visibilidad del uso de los recursos de su cuenta. También puede crear CloudWatch alarmas desde la consola CloudWatch y desde la consola Service Quotas para recibir alertas cuando el uso se acerque a la cuota de servicio aplicada. Para obtener más información, consulte Métricas de uso de Amazon ECR.	28 de febrero de 2020
Actualización de las cuotas de servicio de Amazon ECR	Se han actualizado las cuotas de servicio de Amazon ECR para incluir las cuotas de cada API. Para obtener más información, consulte Cuotas de servicio de Amazon ECR.	19 de febrero de 2020
Comando `get-login-password` agregado	Se ha agregado compatibilidad para get-login-password, que proporciona un método simple y seguro para recuperar un token de autorización. Para obtener más información, consulte Uso de un token de autorización.	4 de febrero de 2020

Cambio	Descripción	Fecha
Escaneo de imágenes	Se ha agregado compatibilidad con el escaneo de imágenes, lo que ayuda a identificar vulnerabilidades de software en las imágenes de contenedor. Amazon ECR utiliza la base de datos de vulnerabilidades y exposiciones comunes (CVEs) del proyecto CoreOS Clair de código abierto y le proporciona una lista de los resultados de los escaneos. Para obtener más información, consulte Escaneo de imágenes para detectar vulnerabilidades en Amazon ECR.	24 de octubre de 2019
Política de punto de enlace de la VPC	Se ha agregado compatibilidad para configurar una política de IAM en los puntos de enlace de la VPC de la interfaz de Amazon ECR. Para obtener más información, consulte Creación de una política de punto de enlace para sus puntos de enlace de la VPC de Amazon ECR.	26 de septiembre de 2019
Mutabilidad de etiquetas de imágenes	Se ha agregado compatibilidad para configurar un repositorio que sea inmutable y evitar que las etiquetas de imagen se sobrescriban. Para obtener más información, consulte Cómo impedir que se sobrescriban las etiquetas de imagen en Amazon ECR.	25 de julio de 2019

Cambio	Descripción	Fecha
Puntos de conexión de VPC de interfaz (AWS PrivateLink)	Se agregó soporte para configurar puntos finales de VPC de interfaz con tecnología. AWS PrivateLink Esto permite crear una conexión privada entre su VPC y Amazon ECR sin que se requiera acceder a través de Internet, a través de una instancia NAT, una conexión de VPN o AWS Direct Connect. Para obtener más información, consulte Puntos de enlace de VPC de la interfaz Amazon ECR ()AWS PrivateLink.	25 de enero de 2019
Etiquetado de recursos	Amazon ECR ha agregado compatibilidad con la adición de etiquetas de metadatos a sus repositorios. Para obtener más información, consulte Etiquetado de un repositorio privado en Amazon ECR.	18 de diciembre de 2018
Cambio de nombre de Amazon ECR	Se cambia el nombre de Amazon Elastic Container Registry (anteriormente Amazon EC2 Container Registry).	21 de noviembre de 2017
Políticas de ciclo de vida	Las políticas de ciclo de vida de Amazon ECR permiten especificar la administración de ciclo de vida de las imágenes de un repositorio. Para obtener más información, consulte Automatice la limpieza de imágenes mediante el uso de políticas de ciclo de vida en Amazon ECR.	11 de octubre de 2017
Amazon ECR admite el manifiesto de imagen de Docker 2, esquema 2	Amazon ECR admite ahora el manifiesto de imagen de Docker V2, esquema 2 (que se usa con Docker, versión 1.10 y posteriores). Para obtener más información, consulte Formatos de manifiesto de imagen de contenedor compatibles con Amazon ECR.	27 de enero de 2017

Cambio	Descripción	Fecha
Disponibilidad general de Amazon ECR	Amazon Elastic Container Registry (Amazon ECR) es un servicio de registro de Docker AWS gestionado que es seguro, escalable y fiable.	21 de diciembre de 2015

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.